Perturbation-Resilient Atomic Commit Protocols for Mobile Environments

Vom Fachbereich Informatik der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erlangung des akademischen Grades eines Doktor-Ingenieur (Dr.-Ing.)
vorgelegt von

Dipl.-Inform. Brahim Ayari

aus Tunis, Tunesien

Referenten:
Prof. Neeraj Suri, Ph.D.
Prof. Philip Koopman, Ph.D.

Datum der Einreichung: 16. Juni 2010
Datum der mündlichen Prüfung: 01. September 2010

Darmstadt 2010
D17

Abstract

The support of distributed atomic transactions is a key requirement for many current mobile applications. Atomicity is a fundamental property ensuring that all nodes reach a consistent outcome. For this, distributed mobile transactions fundamentally require perturbation-resilient atomic commit protocols. This is challenging as mobile environments are typically characterized by frequent network and node perturbations. The environmental constraints on mobile transaction participants and wireless links may increase the resource blocking time of fixed participants. Moreover, frequent node and link failures complicate the design of atomic commit protocols by increasing both the transaction abort rate and resource blocking time. Hence, the deployment of classical commit protocols is not necessarily applicable to distributed mobile environments. Existing protocols ensuring strict atomicity in mobile environments are either bound to very narrow and specific application scenarios or have poor commit rates, high message overhead or a blocking behavior.

In order to cope with different application scenarios, we first identify three classes of mobile environments: *infrastructure-based*, *ad-hoc* and *generic*. Furthermore, we consider and comprehensively classify the perturbations of the wireless mobile environment into classes according to their impact on the outcome of commit protocols and on their blocking behavior. To tolerate these perturbation classes, perturbation-tolerant mechanisms are provided. Based on these mechanisms, we develop a family of perturbation-tolerant atomic commit protocols with minimal resource blocking time and optimized transaction commit rates.

For the *infrastructure-based* mobile environment, we propose an approach that decouples the commit of mobile participants from that of fixed participants – beyond using the strengths of existing approaches. Consequently, the commit set is reduced to a set of entities in the fixed network. Thus, the commit can easily be supported by any traditional atomic commit protocol. For the *ad-hoc* mobile environment, we present a commit approach that supports a significantly wider range of mobility patterns and partitioning scenarios than existing protocols. Our approach is based on a novel coordination strategy using a flexible preselection of multiple coordinators among the participating nodes. Thus, the failure of a single coordinator is tolerated in the presence of network partitioning. For the *generic* mobile environment, we develop an approach that takes advantage of accessing the wired infrastructure if available, by choosing reliable infrastructure nodes for coordinating transactions. If the access to the wired infrastructure is unavailable, our approach adapts itself to the resulting ad-hoc environment.

We evaluated our framework and the algorithms presented in this thesis via extensive simulations and experiments. They validated the efficiency and scalability of the developed solutions and additionally emphasized their resilience to the considered environmental, network and node perturbations by minimizing resource blocking times and optimizing transaction commit rates. Furthermore, they confirmed the suitability of our solutions to a wide range of mobile applications.

Kurzfassung

Die Unterstützung verteilter atomarer Transaktionen ist eine entscheidende Voraussetzung für viele aktuelle mobile Applikationen. Atomizität ist eine grundlegende Eigenschaft, die gewährleistet, dass alle Knoten ein konsistentes Ergebnis erreichen. Um die Atomizität verteilter mobiler Transaktionen gewährleisten zu können, bedarf es störungsresistenter Commit-Protokolle, da mobile Umgebungen typischerweise durch häufige Netzwerk- und Knotenstörungen gekennzeichnet sind. Die umgebungsbedingten Beschränkungen bei mobilen Transaktionsteilnehmern und drahtlosen Verbindungen können die Ressourcen-Blockierungszeit von drahtgebundenen stationären Teilnehmern erhöhen. Darüber hinaus erschweren häufige Knoten- und Verbindungsfehler die Anwendung konventioneller atomarer Commit-Protokollen durch die Erhöhung sowohl der Transaktionsabbruchrate als auch der Ressourcen-Blockierungszeit der Transaktion. Daher ist der Einsatz klassischer Commit-Protokolle für verteilte mobile Umgebungen als nicht empfehlenswert zu erachten. Bestehende Protokolle, die eine strikte Atomizität in mobilen Umgebungen gewährleisten, sind entweder an sehr begrenzte und spezifische Anwendungsszenarien gebunden oder gehen mit niedrigen Commit-Raten, hohem Nachrichten-Overhead oder Blockierungsverhalten einher.

Um die Anwendbarkeit in unterschiedliche Szenarien sicherzustellen, identifizieren wir zunächst drei Klassen von mobilen Umgebungen: Infrastruktur-basiert, ad-hoc und generisch. Des Weiteren klassifizieren wir mögliche Störungen der drahtlosen mobilen Umgebung entsprechend ihrer Auswirkung auf das Ergebnis der Commit-Protokolle und auf deren Blockierungsverhalten. Um diese Störungsklassen zu tolerieren, sind störungstolerante Mechanismen vorgesehen. Basierend auf diesen Mechanismen und unter Berücksichtigung der identifizierten Anwendungsszenarien entwickeln wir eine Familie von störungstoleranten atomaren Commit-Protokollen mit minimaler Ressourcen-Blockierungszeit und optimierten Transaktions-Commit-Raten.

Für die Infrastruktur-basierte mobile Umgebung schlagen wir einen Ansatz vor, der die Transaktionsausführung mobiler Teilnehmer von der stationärer Teilnehmer entkoppelt und der zwar die Stärken bestehender Ansätze subsumiert, dabei aber über die einfache Verwendung dieser Ansätze hinausgeht. Dabei reduziert sich das Commit-Set zu einem Set von Einheiten im Festnetz und es können traditionelle atomare Commit-Protokolle leicht eingesetzt werden. Für die Ad-hoc-Umgebung präsentieren wir einen Ansatz, der eine wesentlich breitere Reihe von Mobilitätsmustern und Partitionierungsszenarien als die bestehenden Protokolle unterstützt. Unser Ansatz basiert auf einer neuen Koordinierungsstrategie mit einer flexiblen Vorauswahl von mehreren Koordinatoren unter den beteiligten Knoten. So kann der Ausfall eines einzigen Koordinators, z.B. bei einer Netzwerk-Partitionierung, toleriert werden. Für die generische Umgebung entwickeln wir einen Ansatz, der den Zugriff auf die drahtgebundene Infrastruktur, falls vorhanden, nutzt, indem zuverlässige Infrastruktur-Knoten für die Koordinierung der Transaktionen ausgesucht werden. Wenn der Zugriff auf die drahtgebundene In-

frastruktur nicht verfügbar ist, passt sich unser Ansatz an die daraus resultierende Ad-hoc-Umgebung an.

Unser entwickeltes Framework und die in dieser Arbeit vorgestellten Algorithmen wurden durch umfangreiche Simulationen und Experimente evaluiert. Die Evaluationsergebnisse demonstrieren die Effizienz und Skalierbarkeit der entwickelten Lösungen und bestätigen ihre Widerstandsfähigkeit gegenüber Netzwerk- und Knoten-Störungen durch Minimierung der Ressourcen-Blockierungszeiten und Optimierung der Transaktions-Commit-Raten. Darüber hinaus zeigen die Ergebnisse die Eignung unserer Lösungen für eine breite Reihe von mobilen Anwendungsszenarien.

Acknowledgements

It is a pleasure to thank all the people who have helped, guided and supported me in the successful completion of this thesis. Without their contributions, this research would not have been possible.

My greatest gratitude goes to my supervisor, *Prof. Dr. Neeraj Suri*, whose encouragement, guidance and support from the initial to the final level enabled me to develop an understanding of the subject. Besides this, he was always patient and helpful whenever his encouragement, supervision and advice were needed. I would like to thank *Dr. Abdelmajid Khelil* for his valuable advice and inspiring discussions on my research work. A great many thanks also to Prof. Philip Koopman for accepting to be my co-advisor.

Then, I would like to thank all past and present DEEDS group's members. Hoping not to forget anyone of the former and current colleges, many thanks to *Faisal, Azad, Hamza, Ripon, Andréas, Dinu, Adina, Robert, Dan, Marco, Peter, Matthias, Piotr, Vinay, Stefan, Daniel, Thorsten* and *Mohammadreza*. Also, special thanks go to *Birgit, Ute* and *Sabine* for helping me with various paperwork and all other circumstances related to living in Germany.

I would also like to thank all my teachers, advisors and friends, who supported me during my education in Tunisia and Germany. They always wished me success and believed in me.

My sincere and heartfelt thanks go to my deceased father, *Abdelwaheb Ayari*, and my mother, *Manana Kharrat VV Ayari*. My mother was always there when I needed her. Her constant moral support has been very encouraging all these years. I also thank my brother, *Saifeddine*, and sisters, *Hager* and *Kaouther*, for their endless love.

I saved the best for the last. I would like to thank my beloved wife, *Emna*, and our beloved daughter, *Eya*, for all the understanding, love, support and happiness that they always brought to me. They were a continuous source of encouragement and strength for me all these years.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction and Problem Context

The pervasiveness and functionality of interacting mobile computing devices is increasing given the progress in wireless technologies. Moreover, computation and storage capabilities of mobile devices, such as laptops, PDAs (Personal Digital Assistants) and mobile phones, are increasing. The mobile wireless systems built using such interacting mobile devices are increasingly playing a major role in our daily life. This statement is validated by statistics on the growth of the mobile device market. It was found that mobile devices equipped with wireless technologies able to connect to the Internet outsell computers (Figure 1.1). In 2009, 450 million such mobile devices were sold, according to Information Week [Information Week, 2009]. Conversely, in 2009, 306 million computers were sold according to Gartner [Gartner]. Since mobile devices already outsell computers, consumers will migrate to handling a significant amount of their daily activities – where computers where earlier involved such as purchasing online products – from mobile devices.

The communication between the interacting mobile computing devices can be either infrastructure-based or ad-hoc and many existing wireless technologies such as WLAN and Bluetooth provide these two types of communication modes:

1. *The infrastructure-based communication mode* allows mobile devices to access wired networks through base stations to communicate with other either mobile or fixed entities.

2. *The ad hoc communication mode* allows mobile devices to communicate directly with each other and in a spontaneous manner when they are in communication range.

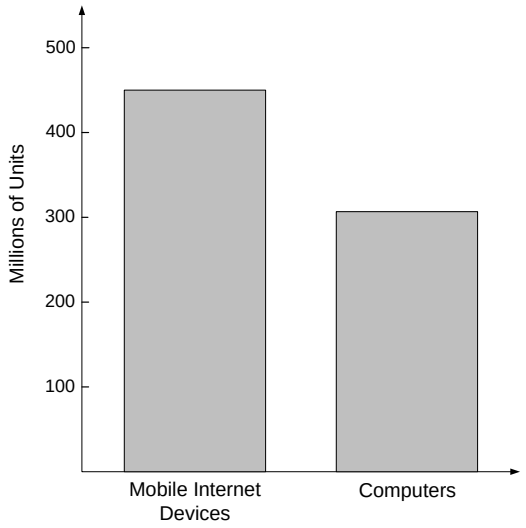

Figure 1.1: Mobile internet devices, computers sold in 2009

Mobile computing devices and base stations are components of *the mobile environment*. When the communication between mobile computing devices is only infrastructure-based, the environment is called *mobile infrastructure-based environment*. In *mobile ad-hoc environments*, mobile computing devices can communicate with each other only in ad-hoc mode. If both communication modes are supported by mobile devices, we are dealing with a *mobile generic environment*.

One of the major purposes of developing wireless systems in mobile environments, especially regarding management of data, is to allow users of such systems to access data *everywhere* and at *any time* and from any type of mobile computing device. Generally, data is stored and managed by databases installed on mobile and fixed devices and accessed through mobile wireless systems. Increasingly, mobile environments are supporting applications that require data consistency. As examples, we provide the following scenarios:

- A user of a mobile device that is able to connect to the Internet can browse online catalogs and buy goods from online shops involving bank and online shop servers in fixed wired networks.

- Mobile game players should be able to see the same state of the game stage at the same time, especially if exchange of virtual or real items is performed during the game.

- A doctor needs to access and update old people's data – living in an am-

bient intelligent [Aarts et al., 2001] environment – independently from his location whether he is in the hospital or outside, visiting another patient.

These mobile environments are characterized by frequent and varied perturbations in the mobile devices and the networks linking them. These are directly apparent as resource constraints and operational failures over the delivery of mobile services as compared to the services delivered by fixed wired networks. Mobile environments are constrained by the processing, storage and energy capacity of mobile devices, and also the continuously varying properties of wireless channels. Most of the failures which occur in such environments manifest at the mobile devices or as communication failures. These failures can last for seconds, minutes or even hours (e.g. network disconnections).

Data in the mobile environment is stored and managed by databases. For a significant range of applications in the mobile environment, such as the ones cited above, data consistency is a key requirement. Preserving data consistency, in distributed database systems in general and in mobile environments in particular, despite the presence of failures relies on the well established *transaction* concept.

A transaction represents a powerful abstraction which encapsulates a set of data operations that should be treated as a single operation. Thus, the transaction has to be executed atomically, i.e., either all the operations are successfully executed and consequently the transaction is *committed*, or none of the operations should affect the data and the transaction is *aborted*. In case the transaction is committed, the effects of the transaction operations become permanent. An aborted transaction has no effects on the data. This "all or nothing" property of transactions is called *atomicity property*. Atomicity is one of the four properties of transactions known as the *ACID properties* [Härder and Reuter, 1983], i.e.,

- **Atomicity:** Transaction modifications must follow the "all or nothing" rule. If one transaction operation fails, the entire transaction fails. It is always critical to maintain the atomic nature of transactions in spite of any failures occurring during the transaction execution.

- **Consistency:** A successful transaction takes the database from one consistent state to another consistent state. If the transaction is not successful, the consistent state of the database before the unsuccessful transaction is restored.

- **Isolation:** If multiple transactions are occurring at the same time, they do not impact each other's execution. The isolation property ensures

that the effects of parallel transactions are equivalent to a sequential execution of these transactions (i.e., without overlapping in time). The isolation property does not guarantee any order of execution of the transactions. It just ensures that they will not interfere with each other.

- **Durability:** If a transaction is committed, the changes performed by this transaction will not be lost. Durability is ensured through the use of database backups and transaction logs that facilitate the restoration of committed transactions in spite of any subsequent software or hardware failures.

In mobile environments, if the data manipulated by a transaction is distributed on different mobile or fixed entities, the transaction is called *distributed* [Breitbart et al., 1992], as opposed to a local transaction which is only executed on either one mobile or fixed entity. In case of local (i.e., non-distributed) transactions, the database itself has to fulfill the atomicity property. However, guaranteeing atomicity in a distributed transaction requires communication among the different entities participating in the execution of the distributed transaction. *Atomic commit protocols* define how these different transaction participant entities should interact with each other in order to reach an atomic decision.

The remainder of this chapter is organized as follows. First, we highlight the problem being addressed by this thesis. Next, the main ideas driving the research in this thesis are presented in the form of research contributions. Finally, we present the structure of the thesis.

1.1 Problem Statement

Transaction atomic commit protocols for wired and fixed networks were devised in the early 1970s and represent a well established and mature research area. However, the frequent and varied perturbations characterizing mobile environments make commit protocols that are designed for fixed wired networks, such as

- the traditional Two-Phase Commit (2PC) protocol [Gray, 1978],
- the Three-Phase Commit (3PC) protocol [Skeen and Stonebraker, 1983], and
- the Paxos Commit protocol [Gray and Lamport, 2006],

unsuitable for mobile environments.

While 2PC is widely applicable in fixed wired networks because the cases in which the protocol blocks are rare, its applicability in mobile environments

is limited as blocking must actually be expected as normal behavior of the system due to frequent failures. 3PC and Paxos Commit solve the blocking problem of 2PC though by adding a considerable message overhead which is often not viable in mobile environments due to considerably higher costs in terms of bandwidth, power consumption, and costs of using wireless links.

Application scenarios where strict atomicity is needed in mobile environments can be divided into three categories:

1. **Mobile infrastructure-based environment scenarios** such as bank/stock transactions. This type of application scenarios includes mobile commerce (m-commerce) applications where users can buy or sell goods using their mobile devices and involving bank and merchant servers in fixed wired networks to accomplish their transactions.

2. **Mobile ad-hoc environment scenarios** such as coordination across autonomous vehicles, mobile gaming and disaster management. Mobile transactions are needed in such type of application scenarios for the purpose of coordination for safe navigation of unmanned autonomous networked vehicles, to guarantee the consistency of game data or to coordinate rescue actions of fire fighters [Obermeier et al., 2009].

3. **Mobile generic environment scenarios** such as health-care ambient intelligence [Aarts et al., 2001] systems. Mobile transaction can help in such type of scenarios to keep data about monitored old people living alone consistent among family members and other institutions such as hospital and insurance e.g. Furthermore, mobile infrastructure-based and ad-hoc environment scenarios can be extended to mobile generic scenarios whenever possible.

Targeting data consistency across all databases involved in such scenarios despite frequent and varied perturbations, the use of atomic mobile distributed transactions is crucial and challenging [Forman and Zahorjan, 1994; Imielinski and Badrinath, 1994; Pitoura and Bhargava, 1994]. Based on the identified application scenarios, the atomic commit problem should not only be tackled in mobile environments in general, but it should be investigated in every identified mobile environment separately. This is due to the fact that these environments show different characteristics and severity of different perturbations which makes the study of everyone of these environments focus on different perturbation aspects than the other one.

In Chapter 2, we show that existing atomic transaction protocols in mobile infrastructure-based environments either guarantee semantic atomicity [Kumar et al., 2002; Serrano-Alvarado et al., 2003] or are based on strict

and hard assumptions not always viable in mobile environments [Bobineau et al., 2000] or does not allow arbitrary disconnections of mobile participant nodes [Nouali et al., 2005]. A comprehensive and systematic perturbation investigation along with its influence on the atomic commit approaches developed for mobile infrastructure-based environments is lacking, and this considerably decreases the attractiveness of these approaches for real world scenario implementations.

In Chapter 2, we also show that existing atomic transaction protocols in mobile ad-hoc environments are either based on consensus [Obermeier et al., 2008], which introduces a considerable message overhead. [Bose et al., 2005; Böttcher et al., 2007] show limited partition-tolerance, because they assume a very specific mobility pattern of mobile nodes which limits the applicability of these solutions. [Xie, 2005] requires global view assumptions such as partition group membership which decreases the applicability of this approach since group membership is hard to realize in typical dynamic mobile ad-hoc environments. Atomic commit approaches developed for mobile ad-hoc environments use distributed system primitives such as consensus and group membership to offer perturbation-resilience. These primitives introduce a significant communication overhead which is not always tolerable in mobile ad-hoc environments.

To the best of our knowledge the problem of transaction atomic commit in mobile generic environments was not investigated in the literature until now.

Each evolving mobile environment (infrastructure-based, ad-hoc and generic mobile environment) requires new commit constraints. Since current approaches are geared towards dedicated scenarios, they do not often provide comprehensive, generalized, efficient and perturbation-resilient commit capabilities. Thus, our research was driven by the need to develop perturbation-resilient and scenario independent commit. One of the major drivers of our research besides perturbation-resilience was efficiency in terms of transaction costs (message complexity, latency, usage of expensive wireless links etc.) and the study of the trade-off between increasing the perturbation-resilience of atomic commit approaches and minimizing the transaction costs (i.e., increasing the efficiency of these approaches).

1.2 Thesis Contributions

The research presented in this thesis makes several important contributions for the mobile transaction research community. Listed below are the main thesis contributions.

(C1) – Perturbation-Resilient Atomic Commit Framework

The major contribution of this thesis is the development of a comprehensive framework of a family of protocols for providing perturbation-resilient transaction commit in emerging mobile environments.

A comprehensive perturbation classification is intended to simplify the design of perturbation-resilient transaction commit protocols in mobile environments by supporting a modular and hierarchical approach. This classification also allows to systematically tackle the problem of atomic commit and present appropriate design techniques to provide resilience to each of the identified perturbation classes without sacrificing performance. The main drivers of defining these new design techniques are *(1)* the minimization of transaction aborts in presence of perturbations and *(2)* the minimization of blocking time of resources especially in the fixed network as fixed participants (e.g. bank servers) are involved in several transactions making them resource critical.

(C2) – Modularity of the Proposed Framework

As mobile systems usually show varied and changing perturbation classes, we stress the modularity of our framework, which simplifies its adaptation at the design stage to different mobile systems by selecting a set of required building blocks implementing the main identified fault-tolerance and recovery techniques. The main building blocks of our work are provided in Chapter 4.

Overall, we combine the developed building blocks into a family of atomic transaction commit protocols for common mobile system classes, i.e., infrastructure-based, ad-hoc and generic mobile systems. The developed protocols are evaluated to highlight the performance/functionality tradeoffs of the different identified building blocks.

(C3) – The FT-PPTC Commit Approach

In mobile infrastructure-based environment, we propose the *Fault-Tolerant Pre-Phase Transaction Commit (FT-PPTC)* approach. The FT-PPTC approach is described in detail in Chapter 5.

The key idea of this approach is to decouple the commit of mobile nodes from that of fixed nodes. Consequently, the execution of the transaction is split into two phases: *(1)* the mobile data gathering phase called *pre-commit phase* collecting "sufficient" information from the mobile nodes to provide progress, and *(2)* the core 2PC phase, which involves only fixed nodes

for commit action. We also develop a comprehensive perturbation-tolerance strategy for the mobile infrastructure-based environment and demonstrate the resilience of the proposed FT-PPTC approach. During the first phase no resources are blocked on fixed nodes because only mobile nodes are involved. Only if the first phase ends successfully, does the second phase start involving fixed nodes. It is important to mention that the FT-PPTC approach does not consider network disconnections as exceptional, but rather as part of the normal operation of the considered system. Our performance studies show that the developed FT-PPTC approach outperforms the existing solutions with respect perturbation-tolerance without sacrificing efficiency. This is achieved by adding a tolerable overhead in terms of message complexity and transaction execution times.

(C4) – The ParTAC Commit Approach

For mobile ad-hoc environments, we propose the Partition-Tolerant Atomic Commit (ParTAC) approach, the first partition-tolerant atomic commit approach for mobile ad-hoc environments which unlike existing approaches, *(a)* does not rely on consensus, *(b)* is independent of the mobility patterns of mobile nodes, *(c)* does not require partition membership knowledge and *(d)* delivers best-effort transactional service availability. The ParTAC approach is explained in detail in Chapter 6.

ParTAC exploits the transaction lifetime concept for mobile transactions in order to reduce transaction decision times. A key idea of this approach is to use multiple coordinators and thus to replicate the coordinator role in order to tolerate unavailability of any subset of coordinators and communication failures. Therefore, ParTAC does not block when some of the coordinators are unavailable for a longer period of time than the lifetime. Furthermore, ParTAC leverages the mobility patterns characteristic for mobile ad-hoc environments by having coordinators collect votes from other participants while moving. These votes are shared and merged once multiple coordinators meet by electing a single coordinator. Our analysis shows that ParTAC reduces the "Commit"/"Abort" decision time of initiated transactions and helps in trading-off the desired level of the availability, latency and efficiency of the transactional service by adapting protocol parameters such as the transaction lifetime and the number of coordinators.

(C5) – The PeRTAC Commit Approach

In mobile generic environments, we propose the Perturbation-Resilient Transaction Atomic Commit (PeRTAC) approach, the first perturbation-resilient

atomic commit protocol for generic mobile environments which combines the advantages of infrastructure-based and infrastructure-less (ad-hoc) solutions. The PeRTAC approach is detailed in Chapter 7.

First, PeRTAC takes advantage of infrastructure-based approaches, if an infrastructure is available, by choosing the more reliable and available fixed nodes to coordinate mobile transactions and to replicate commit data needed to tolerate network and mobile node perturbations. Next, PeRTAC delivers best-effort transactional service availability in case no infrastructure is accessible. Our performance evaluation shows that the PeRTAC approach takes advantage of the access to the infrastructure whenever possible to achieve better performance especially with respect to the transactional service availability by increasing the commit rate of initiated mobile transactions and with respect to commit latency by reducing the commit decision time of these transactions.

1.3 Publications Resulting from the Thesis

The work reported in this thesis is supported by several publications in international conference proceedings and journals:

- **Brahim Ayari**, Abdelmajid Khelil and Neeraj Suri, *On the Design of Perturbation-Resilient Atomic Commit Protocols for Mobile Transactions*, submitted to ACM Transactions on Computer Systems (under revision), 2010

- **Brahim Ayari**, Abdelmajid Khelil and Neeraj Suri, *ParTAC: A Partition-Tolerant Atomic Commit Protocol for MANETs*, in Proceedings of the 11th International Conference on Mobile Data Management (MDM), Kansas City, pp. 135 – 144, 2010

- **Brahim Ayari**, Abdelmajid Khelil, Kamel Saffar and Neeraj Suri, *Demo: Data-based Agreement for Inter-Vehicle Coordination*, in Proceedings of the 11th International Conference on Mobile Data Management (MDM), Kansas City, pp. 279 – 280, 2010

- **Brahim Ayari**, Abdelmajid Khelil and Neeraj Suri, *Exploring Delay-Aware Transactions in Heterogenous Mobile Environments*, in Journal of Software - Special Issue: Selected Papers of The 6th IFIP Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, vol. 4, no. 7, Academy Publisher, Finland, pp. 634 – 643, 2009

- **Brahim Ayari**, Abdelmajid Khelil and Neeraj Suri, *Delay-Aware Mobile Transactions*, in Proceedings of the 6th IFIP Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS), Capri Island, Italy, pp. 280 – 291, 2008

- **Brahim Ayari**, Abdelmajid Khelil, Neeraj Suri and Eugen Bleim, *Implementation and Evaluation of Delay-Aware and Fault-Tolerant Mobile Transactions*, in Proceedings of the 2nd International Conference on E-Medical Systems (E-MediSys), Sfax, Tunisia, 2008

- **Brahim Ayari**, Abdelmajid Khelil and Neeraj Suri, *FT-PPTC: An Efficient and Fault-Tolerant Commit Protocol for Mobile Environments*, in Proceedings of the 25th IEEE Symposium on Reliable Distributed Systems (SRDS), Leeds, UK, pp. 96 – 105, 2006

Besides the work presented in this thesis, the author has been involved in publications related to static and mobile wireless sensor networks [Khelil et al., 2008, 2009, 2010; Shaikh et al., 2010].

1.4 Thesis Structure

The structure of the following chapters is illustrated in Figure 1.2. It follows the structure of the thesis contributions described earlier:

Chapter 1 presents the background of the problems driving this research, introduces the research problems and the contributions of this thesis.

Chapter 2 surveys the state of the art and practice in the field of atomic commit protocols especially in mobile environments.

Chapter 3 presents and discusses the system model used throughout in this thesis. Subsequently, the transaction and perturbation models are presented.

Chapter 4 introduces our perturbation-resilient atomic commit framework. First, we investigate the design requirements of resilient atomic commit protocols in mobile environments. Next, we present perturbation-resilient design techniques relevant to each perturbation class identified in Chapter 3 and driven by our design requirements.

Chapter 5 introduces our solutions towards perturbation-resilient atomic commit in mobile infrastructure-based environments. The correctness of the presented solutions is proven and a simulation-based evaluation is provided.

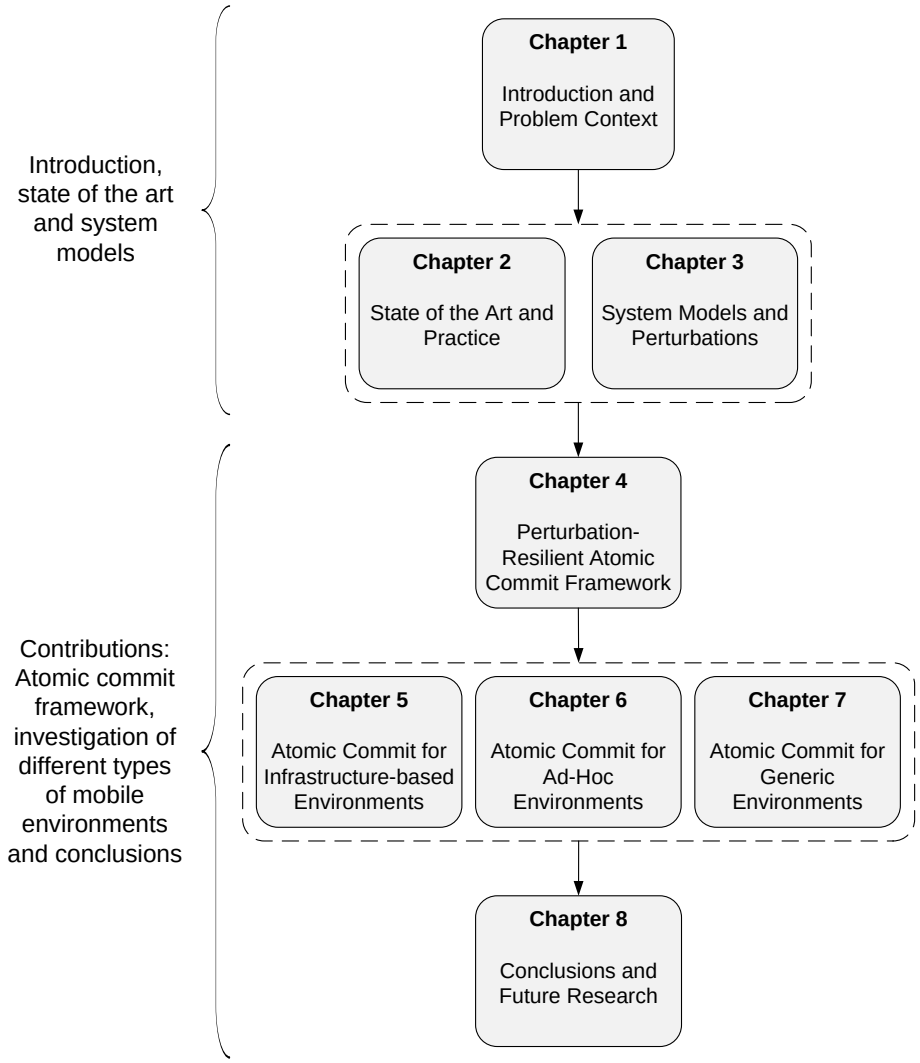

Figure 1.2: Thesis structure

Chapter 6 presents our partition-tolerant atomic commit solution for mobile ad-hoc environments. In this chapter, we provide also a correctness proof of the presented solution and a simulation-based evaluation taking into account all important parameters of the investigated ad-hoc environment.

Chapter 7 describes the perturbation-tolerant atomic commit protocol for mobile generic environments. Similar to chapters 5 and 6, we prove the correctness of our approach and use simulations to evaluate its performance by varying all relevant parameters of the mobile generic environment.

Chapter 8 finally concludes the thesis, re-evaluating the value of the conceptual contributions. A discussion on the applicability of the thesis results to mobile environment scenarios is provided alongside with an outline of the future research directions opened by the novel approaches presented by this thesis.

Chapter 2

State of the Art and Practice

As an important basis for the context of the research presented in the thesis, this chapter starts by discussing the different existing transaction commit protocols. First classical transaction commit protocols designed for wired networks are described. Next, atomic commit protocols developed for mobile infrastructure-based environments are listed. Each protocol is described in detail along with its strengths and weaknesses. Finally, existing atomic commit solutions for mobile ad-hoc environments are presented along with a discussion of their advantages and disadvantages in the mobile environment they are designed to work in.

This chapter forms the background and the context for the research questions posed and puts the contributions presented into perspective. The chapter concludes with a discussion on the need for perturbation-resilient atomic commit in mobile environments.

2.1 Classical Transaction Commit Protocols

We describe in this section the two classical atomic commit protocols: Two-Phase Commit (2PC) and Three-Phase Commit (3PC). 2PC and 3PC are mainly used within fixed environments where the communication is reliable as compared to mobile environments. Furthermore, we briefly present the Paxos Commit Protocol that solves the atomic commit problem based on the Paxos Consensus algorithm.

2.1.1 Two Phase Commit

The Two-Phase Commit Protocol (2PC) [Gray, 1978] represents the first atomic commit for distributed transactions. 2PC is a two phases atomic commit protocol as its name implies. Figure 2.1 illustrates these two phases of the protocol. The first phase is a votes collection phase where the coordinator collects votes from each transaction participant. Each participant can vote either to commit the transaction ("Yes" vote) or to abort it ("No" vote). After collecting all votes, the coordinator starts the second phase. The second phase is a decision phase in which the coordinator either decides to commit the transaction if all collected votes are "Yes" or to abort it otherwise.

Any participant that voted "Yes", i.e. to commit the transaction, has to wait until it receives the decision from the coordinator and then it can, based on the received decision, either commit or abort the transaction. When a participant votes "No", i.e. to abort the transaction, it can immediately abort the transaction without waiting for the decision of the coordinator. In this case all the changes caused by the aborted transaction must be rolled back. In case a participant voted "Yes" and the decision sent by the coordinator is "Commit" the changes of the transaction become permanent. Only at this point in time the transaction's changes can be made visible to other transactions. Furthermore, locks held by the completed transaction can be released after the completion of all write operations performed by this transaction. After receiving the coordinator's decision, each participant sends an acknowledgement to the coordinator.

In order to deal with communication and system failures, 2PC uses a logging schema where every message is written to the stable storage before being sent to its recipient. Note that the message is only sent when the logging process ends, i.e., when the writing of the message to the stable storage finishes. For n participants, the classical 2PC protocol exchanges $4n$ messages. There exists also further optimizations of the 2PC protocol such as the Presumed Commit and the Presumed Abort protocols [Mohan and Lindsay, 1985].

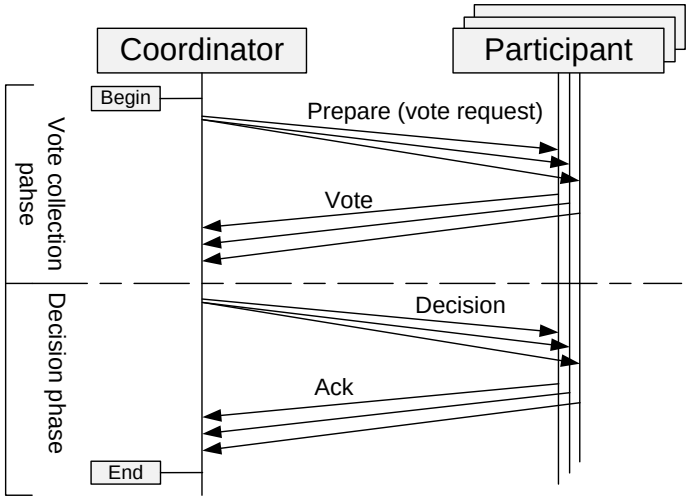

Figure 2.1: The 2 phase commit protocol

2.1.2 Three Phase Commit

The major problem of 2PC and its optimizations is that 2PC involves a waiting state. Participants enter the waiting state after they have sent their votes to the transaction coordinator and remain in this state until they receive the decision from the coordinator. In 2PC, both commit and abort states can be reached directly from the waiting state. A coordinator failure combined with a participant failure (the coordinator is in general the transaction initiator and therefore a transaction participant) can lead to a blocking situation in which the remaining participants cannot decide for abort or commit, since the failed participant might have reached either the abort or the commit state. Even a new elected coordinator is not able to terminate the transaction when the state of at least one participant is unknown. This blocking behavior of 2PC makes it unsuitable for mobile environments where mobile node failures are relatively frequent as compared to fixed node failures.

The Three-Phase Commit Protocol (3PC) [Skeen and Stonebraker, 1983] was developed to overcome this blocking problem of 2PC in environments where site failures (not network partitioning) can occur. 3PC introduces an additional dissemination phase to 2PC before the decision phase. Figure 2.2 illustrates a successful execution of the 3PC protocol. During the first phase of 3PC, the coordinator collects the votes of the participants. If and only if all received votes were “Yes”, the coordinator sends a “PrepareToCommit” message to each participant. A participant that has received the “Prepare-

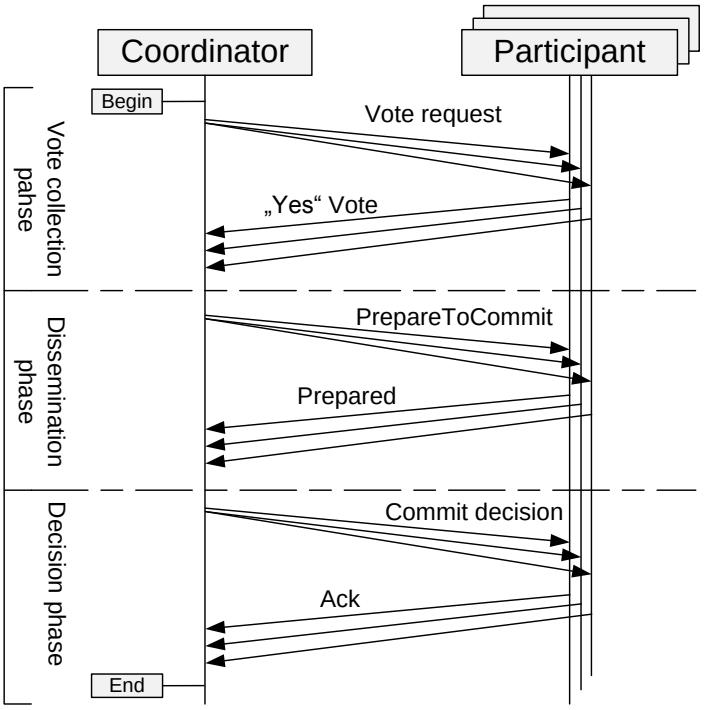

Figure 2.2: The 3 phase commit protocol

ToCommit" message acknowledges by sending "Prepared". The coordinator needs to receive the "Prepared" messages of all participants to decide to commit the transaction.

Note that a participant that has sent a "Yes" vote is not allowed to change its vote and abort the transaction on its own. At first, the "PrepareToCommit" and "Prepared" messages seem to be superfluous, but by adding this new phase, 3PC guarantees that for any two participants which may differ in at most one state, the set of reachable states does not contain an abort and commit states simultaneously. If the coordinator fails, the participants can elect a new coordinator that terminates the transaction. This new coordinator collects from all available participants their last state within the protocol execution. If at least one participant has received a "PrepareToCommit" message, the new coordinator can safely decides to commit the transaction. In this case, the new coordinator knows that all participants have voted "Yes", and no participant may have voted "No". If no participant received a "PrepareToCommit" message, no participant can have committed the transaction. Thus, the new coordinator will decide to abort it.

It is worthy to mention at this stage that the termination phase of 3PC

is correct only when network partitioning cannot occur. In case of network partitioning, two new coordinators might be elected in two different network partitions and may take different decisions about the outcome of the transaction. Because mobile networks are susceptible to network partitioning in general, the termination mechanism of 3PC can lead to inconsistent transaction outcomes.

2.1.3 Paxos Commit

The Paxos Commit Protocol [Gray and Lamport, 2006] is based on the Paxos Consensus approach introduced in [Lamport, 1998; Lamport and Massa, 2004]. The Paxos approach was originally designed to solve the consensus problem, i.e., to let participants agree on a single decision among several possible outcomes. Paxos Commit uses multiple coordinators and assigns to them different roles. One of them is called the leader, the other coordinators are called acceptors. Each participant sends its vote to all Paxos acceptors. Each acceptor collects the received votes and then forwards them to the leader. The leader decides about the outcome of the transaction. If some acceptors have not received some of the votes, the leader decides whether some acceptors must be additionally notified. When the leader has made a proposal, it is sent to the acceptors and to the participants.

To guarantee that the protocol makes progress even if the leader fails, each acceptor has the ability to decide for itself if and when it becomes a leader. To resolve conflicts in case more than one leader is present, Paxos uses increasing version numbers to identify the highest leader and the corresponding leader's proposal. An acceptor accepts a new proposal only if the new proposal has a higher version number than the old proposal. A new leader must build a proposal by adopting the previous proposal having the highest version number. In contrast to the termination phase of 3PC, Paxos Consensus works correctly even when network partitioning occurs. The major disadvantage of Paxos Commit in mobile environments is its high message overhead which is often not viable in mobile environments due to considerably higher costs in terms of bandwidth, power consumption, and charges of using wireless links.

2.2 Commit Protocols for Mobile Infrastructure-Based Environments

Given the need for correct data management in infrastructure-based mobile environments, mobile transactions have increasingly become the focus of

extensive ongoing research. A variety of transaction models have been proposed such as [Alonso and Korth, 1993; Chrysanthis, 1993; Dunham et al., 1997; Gray et al., 1998; Ku and Kim, 2000; Madria and Bhargava, 2001; Pitoura and Bhargava, 1995, 1999; Walborn and Chrysanthis, 1995, 1999; Yeo and Zaslavsky, 1994] with an excellent survey appearing in [Serrano-Alvarado et al., 2004a]. Some transaction models such as [Karlsen, 2003; Nouali-Taboudjemat and Drias, 2008; Serrano-Alvarado et al., 2005] propose adaptability to different mobile environments, constraints and applications by relaxing the traditional ACID properties (Atomicity, Consistency, Isolation and Durability) which leads to temporary inconsistent states. Some commit protocols have recently started addressing the problem of atomic commit in mobile infrastructure-based environments [Bobineau et al., 2000; Kumar et al., 2002; Nouali et al., 2005; Serrano-Alvarado, 2004]. The performance of these protocols and additionally 2PC are evaluated in [Bobineau et al., 2004; Nouali-Taboudjemat et al., 2007]. In this section, we describe atomic commit protocols developed for mobile infrastructure-based environments.

2.2.1 Unilateral Commit for Mobile

The work on Unilateral Commit for Mobile (*UCM*) [Bobineau et al., 2000] provides support for disconnections and off-line executions on mobile devices. UCM is a one-phase protocol where the voting phase of the 2PC is eliminated by enforcing some properties on the participant's behavior during the transaction execution. UCM reduces the atomic commit protocol to a single phase that consists in broadcasting the coordinator's decision to all participants. In other words, the coordinator acts as a "dictator" imposing its decision on all participants. The coordinator is also chosen in UCM to be a fixed node. If a participant is not able to conform to this decision due to a crash, the coordinator simply forward-recovers the corresponding transaction branch.

UCM is a 1 phase commit (1PC) protocol which introduces the following assumptions on the way participants manage transactions [Bobineau et al., 2000]:

- 1PC protocols assume that all the transaction operations have been acknowledged (i.e., have been successfully executed till completion) before the protocol is launched. This means that the Atomicity of all the local transaction branches is already ensured at commit time.

- 1PC protocols assume that integrity constraints are checked immediately after each update operation and before acknowledging the opera-

tion. Thus, Consistency is ensured for all the local transaction branches at commit time (no deferred integrity constraints).

- 1PC protocols assume that all participants serialize their transactions using a pessimistic concurrency control protocol that avoids cascading aborts (i.e., strict two-phase locking [Bernstein et al., 1987]). This actually means that serializability (Isolation) of all the local transaction branches is already ensured at commit time.

- 1PC protocols assume that, at commit time, the effects of all the local transaction branches are logged on stable storage, and hence the Durability property is ensured. This means that the log on the coordinator contains all the transaction updates and that this log is forced on disk before the 1PC is launched. The content of this log and the way it is exploited at recovery time differ strongly among the protocols.

UCM has been specifically designed for mobile environments. To guarantee atomicity, transaction operations and their acknowledgments are continuously logged. If a problem arises the global transaction is immediately aborted. Therefore, if a global transaction reaches the validation phase the global decision is commit. There is no risk of inaccurate global abort. A global commit is performed in a single phase, the decision phase. It is initiated by transferring the transaction's operation log from the application which initiated the transaction to the coordinator. Transaction participants are represented by proxies to interact with the UCM coordinator. This choice improves the participant's autonomy and insures proper recovery in case of participant failure.

UCM guarantees strict atomicity and reduces the wireless message complexity. However, UCM is based on strict and hard assumptions such as local pessimistic concurrency control (strict two-phase locking) which is required for all participants, as well as immediate integrity control and homogeneity of participating database systems (we refer to the 1PC assumptions above). These assumptions restrict also the applicability of UCM to only a subset of the possible applications in mobile infrastructure-based environments. Similar to 2PC, an UCM coordinator blocks in the waiting state if at least one acknowledgement message is missing.

2.2.2 Transaction Commit On Timeout

Transaction Commit On Timeout ($TCOT$) [Kumar et al., 2002] uses timeouts to provide a non-blocking protocol that limits the amount of communication

between the participants in the execution of the protocol. Instead of exchanging messages to reach a Commit or Abort decision, the coordinator waits for timeouts to expire. In TCOT, if the coordinator node does not receive a failure message from a participant within a predefined timeout period, then it commits the transaction. TCOT defines two types of timeout: Execution Timeout and Update Shipping Timeout.

- Execution Timeout (E_t) defines an upper bound timeout value within which a transaction participant completes the execution (not commit) of its transaction operations. The value of E_t is in general node specific. It might depend on the number and complexity of transaction operations and computing capabilities of the transaction participant. Additionally to the time needed to execute the transaction operations on the database itself, E_t accounts for the characteristics of mobile devices such as poor resources, disconnected state, availability of wireless channel, etc. It is possible that a mobile device while executing TCOT may take more or less time than its E_t to execute its transaction operations. In TCOT, E_t typically should be just long enough to allow the transaction operations to be successfully executed in a normal environment (i.e., no failure of any kind, no message delay, etc.)

- Shipping timeout (S_t) defines the upper bound of the data shipping time from the participant to the coordinator. Thus, at the end of E_t, the coordinator expects the updates to be shipped to it and logged there within S_t. S_t is computed as Time to compose updates + Time for the updates to reach the coordinator.

While processing its transaction operations, if a participant finds out that its operations will execute longer than E_t, then it extends its value and sends it to the coordinator. If a participant decides to abort for any reason, then it sends a "No" vote to the coordinator before the expiration of E_t. If the participant successfully executes its transaction operations, it sends a log of the updates it made to its local database copy to the coordinator. The updates must reach to coordinator before the expiration of S_t. It is possible that updates reach the coordinator much earlier, in which case it will decide to commit before the expiration of its timeouts. Once the updates are shipped to the coordinator, the participant commits the transaction locally. In case the participant mobile node fails to send its updates to the coordinator within S_t and it did not extend E_t, then the coordinator decides to abort the transaction.

In TCOT the coordinator is implemented on the base stations. The base station to which the initiator of the transaction is connected plays the

coordinator role. If the transaction initiator moves to an area covered by another base station, the information stored by the coordinator about the execution state of the transaction moves also to the new base station during the hand-off process.

Overall, TCOT provides only semantic atomicity as defined in [Garcï¿½a-Molina, 1983]. Semantic atomicity requires the existence of a compensating transaction for every initiated mobile transaction which is not possible for every transaction. Compensating transactions undo semantically the transaction effects. This type of atomicity is weaker than the strict atomicity [Härder and Reuter, 1994] needed for transactions in general, which limits the applicability of TCOT to a limited class of applications.

2.2.3 Combination of Optimistic Approach and 2PC

The CO2PC protocol [Serrano-Alvarado, 2004; Serrano-Alvarado et al., 2003] combines an optimistic approach with 2PC. Like TCOT, the objective of the CO2PC protocol is to provide semantic atomicity for execution alternatives by allowing participants to perform either optimistic local commit (locally committed results are shared) or non-optimistic commit. Semantic atomicity limits the applicability of the protocol only for a restricted set of applications. The authors relax strict atomicity in this work (by guaranteeing only semantic atomicity) to increase the flexibility of participants (particularly mobile nodes). Hence, they distinguish between compensable transactions that are committed locally in an optimistic manner and non-compensable ones that have to wait for the global decision.

The CO2PC protocol works as follows: all participants send a vote ("Yes"/"No") to the coordinator which takes the global decision and sends it to all participants. A unanimous "Yes" vote leads to a commit decision, otherwise the decision is abort. The CO2PC coordinator is chosen to be participating fixed node in the transaction execution. If there is no fixed node participant then the base station will play this role. Mobile nodes are not chosen to play the coordinator role because they are subject to frequent disconnections and have restrained resources for logging and communication.

A participant making optimistic commit (called optimistic participant) executes its transaction operations and commits/aborts the transaction unilaterally. Then, it sends its vote to the coordinator. If the global decision is commit, optimistic participants are done; if it is abort they have to launch compensating transactions. Once compensating transactions are initiated they should complete successfully. A participant making non-optimistic commit (called non-optimistic participant) executes the 2PC protocol locally. The 2PC part of the protocol is made on the same node and does not require

messages through the wireless network.

2PC is used by CO2PC because – for non-compensable transactions – resources must be retained until a global commit/abort. 2PC provides this state. Moreover, it ensures atomicity, produces recoverable systems, avoids cascading aborts (when it is combined with two-phase locking) and its interface is widely implemented. 2PC is not used for the global coordination of transaction atomicity.

To provide recovery, CO2PC records its progressing steps in the coordinator and participant logs. Since failures and disconnections can occur at any moment, logging information should be forced to be written (i.e. flushed into a stable storage) before sending messages. For mobile nodes, CO2PC information will be logged in base stations. The authors do not consider physical storage of mobile nodes to be stable because mobile nodes are subject to loss, damage and undefined disconnections.

2.2.4 Mobile Two-Phase Commit

The objective of the Mobile Two-Phase Commit (*M-2PC*) protocol [Nouali et al., 2005] is to globally commit a distributed transaction in a mobile environment. M-2PC assumes that a transaction is issued at a mobile node. While a mobile node moves from a cell (geographical area covered by a base station) to another cell; it connects to a new base station. The M-2PC protocol can either terminate in the same cell where it is initiated or in a new cell covered by a new base station. Similarly to the 2PC protocol, there are three important roles a node can play in M-2PC: *the transaction initiator role* played by the mobile node launching the transaction, *the participant role* played by the processing entities of the transaction operations and *the coordinator role* played by the entity coordinating the consistent termination of the transaction.

The strategy chosen in M-2PC is to split the duties of M-2PC protocol into two tasks: The first one maintains the same schema on the fixed part of the network as in traditional 2PC; the second one adjusts the schema to manage the mobile wireless part. The coordination of the fixed nodes decision is conducted in M-2PC as it is in the classical 2PC protocol. In M-2PC, the coordinator is chosen to a fixed node in order to be directly reachable by the participant fixed nodes. The coordinator should also be reachable by mobile nodes. Therefore, the coordinator resides in M-2PC on the base station of the transaction initiator mobile node. This implies that the coordinator in M-2PC is likely to be located as close as possible to the initiator mobile node. The coordinator executes on the same base station even if the mobile node moves to another cell during the execution of the commit protocol.

In order to deal with disconnections, M-2PC lets the mobile node delegate its commit duties to the coordinator which is assumed to be always available during the protocol execution. The mobile node sends the request for commit to the coordinator along with its logs. The mobile node can then disconnect. The coordinator sends vote request messages to all participants and decides on whether to commit or abort according to the classical 2PC semantics. Once the coordinator receives the acknowledgements of all participants, it informs the initiator about the result. The coordinator waits for the client acknowledgement before forgetting about the transaction (by releasing all resources acquired by the transaction).

M-2PC is also one of the first atomic commit protocols to consider scenarios where at least one server or participant (other than the transaction initiator) is mobile, i.e., the transaction accesses, for example, data located on a mobile node. In [Nouali et al., 2005], the following situation is given as an example scenario: a researcher meets other researchers at a conference and needs to agree on a rendezvous with these researchers. In this application, the researchers' respective agendas have to be synchronized and therefore all the participants are mobile. To mange such scenarios, the M-2PC protocol proposes to have in the mobile participant side a scheme similar to that of the initiator side. A representation agent will work on behalf of the mobile server which is free to disconnect starting from the moment it delegates its commitment duties to its representation agent. This agent is responsible of transmitting the result to the participant at reconnection time and also of keeping logs and eventually recovering in the case of failure. By adopting this schema M-2PC shifts the workload to the fixed part of the network in order to preserve valuable processing power and communication resources and also to minimize the cost of using wireless links.

To summarize, in M2PC, a mobile participant delegates its commit duties to its agent on a fixed node, which is the base station the mobile participant is connected to. Unfortunately, M-2PC assumes that all mobile participants are connected at transaction initiation and that network disconnections are allowed only after the mobile node delegates its commitment duties.

2.3 Commit Protocols for Mobile Ad-Hoc Environments

We outline in this section existing atomic commit protocols developed for mobile ad-hoc environments along with their advantages and limitations in mobile ad-hoc environments.

2.3.1 Cross Layer Commit

In [Obermeier et al., 2008], a cross layer commit protocol for mobile ad-hoc networks (CLCP) is presented. CLCP is based on the Paxos Commit approach. According to [Obermeier et al., 2008], Paxos Commit is one of the best protocols regarding failure tolerance. Paxos Commit uses multiple coordinators and allows even half of them to fail during protocol execution. This failure tolerance is optimal as described in [Skeen and Stonebraker, 1983]. However, the authors in [Obermeier et al., 2008] Paxos Consensus identified the following problems when it is used in mobile ad-hoc networks:

- Although it uses multiple coordinators, Paxos Commit is a centralized protocol, i.e., a special leader is still necessary. Each participant may decide for itself during protocol execution if and when it becomes a leader. Even though [Gray and Lamport, 2006] proposed an additional decentralized variant for a faster commit, this variant does not allow the protocol to terminate in a decentralized way if a database's vote message is lost or delayed.

- The number of messages increases with the number of leaders, since each message is routed to the special leader.

- As the experiments performed by the authors in [Obermeier et al., 2008] have shown, the performance of Paxos Commit is highly dependent on the use of acknowledgement messages. Without acknowledgements, the performance degrades significantly.

- It is an application layer protocol: When acknowledgements are used, the protocol is not designed to make any use of these acknowledgements, i.e., it does not use them for gaining global knowledge.

CLCP assumes that each participant of a global transaction knows all participants in the atomic commit protocol. CLCP consists of two phases. The first phase is called the decentralized commit phase. During this phase, the participants vote and concurrently try to come to a decentralized commit decision. Each participant sends its vote for the transaction to all of its neighbors within a single transmission, forwards received votes to all of its neighbors, and determines by CLCP if and when the transaction must be committed or aborted. If a participant does not send any vote at all, CLCP can also abort the transaction within the decentralized commit phase without requiring a centralized leader.

In [Obermeier et al., 2008], the authors claim that the case where the protocol cannot progress due to network partitioning is seldom. In this case, a

termination phase follows the decentralized commit phase. During the termination phase, one participant becomes a special participant called leader that organizes the commit decision and ensures that a majority of participants, i.e., more than 50% of all transaction participants, accept this decision. If the leader fails or the commit decision cannot be made after a timeout, a different participant becomes a new leader having an increased version number that identifies it as the new leader. According to the experimental evaluation made in [Obermeier et al., 2008], the transaction decision is made within the decentralized commit phase in most cases. The termination phase uses an extension of the Paxos Commit algorithm [Gray and Lamport, 2006] to identify a participant as a leader that is allowed to form a proposal. However, in contrast to [Gray and Lamport, 2006], a distributed termination algorithm that contains only one centralized step is used in CLCP. This step is the determination of a new proposal.

To summarize, the CLCP protocol employs all participants as coordinators and uses a consensus-like approach to ensure failure tolerance. CLCP is directly instantiated from the application layer, but operates on both network and application layers. Consensus introduces a considerable message overhead to mobile ad-hoc environments which makes it undesirable in mobile ad-hoc networks.

2.3.2 Integrated Commit

In [Bose et al., 2005; Böttcher et al., 2007], the authors propose the use of a *cluster of coordinators* preferably in single-hop distance from each other to avoid blocking of participating mobile nodes in case one coordinator fails. While communication with the participants is done using the classical 2PC protocol, the cluster of coordinators uses the 3PC protocol to eliminate the blocking behavior of 2PC in case of coordinator failure. The cluster of coordinators elects a single *main coordinator* and uses the 3PC protocol to agree on a consistent decision either to commit or abort the transaction. The major assumption of this protocol lies in the fact that the cluster of coordinators is located in an environment with fast message transfer, for example, if this cluster of coordinators is situated in single-hop distance for each other. Additionally, if integrated commit assumes that network partitioning cannot split the cluster of coordinators, the protocol can proceed without blocking the coordinators even if some of them fail.

In the failure-free case, the protocol works as follows: The transaction execution is initiated by an initiator which could be a separate node or a database itself. A transaction in integrated commit is divided by the initiator into subtransactions and these subtransactions are sent to the appropriate

participants. Each participant executes its corresponding subtransaction until it can decide whether to send a "Yes" or "No" vote. Then this vote is sent to its associated coordinator. Each associated coordinator starts a timer and accepts votes from other participants for a certain period of time. When the specified timeout has passed, the coordinator will no longer accept any vote. The coordinators then bundle their collected "Yes"/"No" votes and forward them to the main coordinator. After a specified period of time, the main coordinator decides on a global Commit/Abort and informs the coordinators. The main coordinator's decision is to commit the transaction if all votes were "Yes" and is to abort otherwise.

The coordinators wait for a "prepare-to-forward-commit message" from the main coordinator, and upon receiving this message, they send an acknowledgment message to the main coordinator. After having received all acknowledgment messages, the main coordinator sends a "commit valid" message to the coordinators. Each coordinator forwards the decision to its associated participant. In the integrated commit protocol the whole cluster of coordinators acts as a commit coordinator. The main coordinator acts as the commit coordinator and the individual coordinators act as the participants in the 3PC protocol. The transaction participants act as the participants in the 2PC protocol.

If the cluster of coordinators is partitioned or the main coordinator fails the authors use a termination protocol based on the Paxos consensus protocol [Lamport, 1998] to elect a new main coordinator. The assumption in [Bose et al., 2005; Böttcher et al., 2007] that the coordinators are moving together in a group (forming a one hop cluster e.g.) is not valid in most of ad-hoc scenarios. Targeting a more generic solution, this assumption on the mobility pattern of a subset of the mobile nodes in the mobile ad-hoc environment needs to be relaxed to consider a generalized arbitrary mobility model.

2.3.3 Group Based Transaction Commit

In [Xie, 2005] a commit solution is presented which assumes that every mobile node in a partition knows all the members of the partition it belongs to. The group based commit protocol is based on the following assumptions:

- Partitions can be detected.
- An asynchronous path exists between any two members. An asynchronous path exists if two nodes are not in the same partition at the same time, however, they are able to communicate through other nodes passing the information between them at different times.

- Each site maintains a log.
- A central coordinator is not always available.
- Participants of a transaction are decided at the beginning of the transaction and will not be changed.

Because a central coordinator is not assumed, group based transaction commit needs to find a group coordinator for each partition. A group coordinator can be elected randomly from the set of participants present in a partition or by applying a utility function and find the optimal participant to play the group coordinator role. For example, the utility function could be the cost of communicating with other participants in the partition. Network partitioning divides the transaction participants into different groups (participants present in one partition). In the group based commit protocol, the transaction can still make the decision even if all participants are not connected into a single partition again. The basic idea of the protocol is to utilize the frequent membership change within different partitions. When a participant changes the membership from one group to another group, it could carry the transaction states in the old group into the new group. So the participants in the new group will know the decisions made in the old group and will be able to make the proper decision.

Given the *partition membership information*, every partition in the group based transaction commit elects a leader and uses the 2PC protocol inside the partition to decide whether the transaction should be tentatively committed or aborted. This temporary decision is communicated to all participant mobile nodes within the partition. When a participant mobile node joins a new partition, the tentative decision (obtained in its original partition) is communicated to the new partition.

As described in [Xie, 2005], the correctness of the proposed solution is assured by the partition membership assumption, i.e., the fact that partitions can be detected. The assumption that every mobile node in a partition knows all the members of its partition is crucial for a generalized mobile ad-hoc environment. Some works [Briesemeister and Hommel, 2002; Roman et al., 2001] addressed the problem of group membership in mobile ad-hoc environments. However, a generic solution remains a challenge. Furthermore, the blocking time of participant mobile nodes is often not considered and in worst case all participant mobile nodes may be blocked forever if one of the participant mobile nodes disconnects. As shown in [Skeen, 1981], there exists no non-blocking atomic commit protocol if network partitioning may occur for an unpredictable duration. But at least this blocking behavior should

be studied and solutions should be proposed which minimizes the blocking times of transaction participants.

The approach briefly described in [Xie, 2005] and based on partition membership information does not use consensus and is independent from the mobility of nodes in contrast to [Bose et al., 2005; Böttcher et al., 2007; Obermeier et al., 2008]. However, it is based on the assumption that partition membership is available to all its members. Partitions in mobile ad-hoc environments are usually very dynamic as nodes may leave and join partitions arbitrarily. Therefore, acquiring the global partition membership information becomes very inefficient.

Table 2.1 summarizes the main advantages and disadvantages of existing atomic commit protocols in ad-hoc mobile environments.

Table 2.1: Commit protocols for mobile ad-hoc environments

Protocol	Uses consensus	Uses partition/ group membership	Requires specific mobility pattern
[Obermeier et al., 2008]	Yes	No	No
[Bose et al., 2005] [Böttcher et al., 2007]	Yes	No	Yes
[Xie, 2005]	No	Yes	No

2.4 Chapter Summary

As each evolving mobile environment (infrastructure-based, ad-hoc and generic mobile environment) necessitates new commit constraints, the current approaches geared towards dedicated scenarios, often do not provide comprehensive, generalized, and perturbation-resilient commit capabilities. Thus the need to develop perturbation-resilient and scenario independent commit drives our research.

Our goal is to investigate how to comprehensively introduce perturbation-resilience and scenario independency to all the identified types of mobile environments, i.e., the mobile infrastructure-based environment, the mobile ad-hoc environment and the mobile generic environment. This investigation will take place in Chapters 5, 6 and 7 respectively.

It is important to mention in this chapter that, to the best of our knowledge, no work before this work investigated atomic commit protocols in generic mobile environments.

Chapter 3

Mobile Environment Models

The perturbation-resilient atomic commit framework developed in this thesis is based on the mobile environment models presented in this chapter. The mobile environment models mainly consist of the system model, the transaction model and the perturbation model. The system model describes the different components of the considered mobile environment. The system model also provides a classification of the mobile environment into sub-environments based on the different application scenarios available and possible in the considered mobile environment. The transaction model establishes the formal definition of a mobile transaction and the terminology related to it and used throughout this work. Finally, the perturbation model is presented. The perturbation model covers the various environmental constraints and failure modes that can be encountered in the stated system model. All these models and especially the perturbation model are comprehensively taken into consideration in the design issues presented in the next chapters.

3.1 System Model

In order to consider and support a broad class of mobile applications, we consider a generalized mobile distributed environment consisting of sets of battery-powered mobile nodes (MNs) and fixed nodes (FNs). We refer to the set of MNs and FNs as $M = \{MN_1, \ldots, MN_m\}$ and $S = \{FN_1, \ldots, FN_s\}$ respectively, where m and s are the number of MNs and FNs respectively. The different components of the mobile environment are illustrated in Figure 3.1. To keep our mobile environment general, we consider also the existence of mobile sinks (MSs) which collect data from a set of sensor nodes (SNs) about the environment being monitored. Note that SNs are only mentioned in our system model in order to support a wide range of applications and not as active participants in transactions. Wireless sensor networks are interpreted in our work as a database which can be queried to retrieve information about monitored area or goods. We assume that every node except SNs have a unique ID in the considered generalized environment.

Some MNs are equipped with appropriate wireless interfaces and can intermittently connect to the wired network through base stations (BSs) via wireless channels. These MNs can communicate with each other or with FNs using the services provided by BSs. A subset of the MNs can communicate directly or multihop with each other in an ad-hoc manner for instance using Bluetooth, the ad-hoc mode of WLAN etc. Every MN has at least one wireless interface and is able either to communicate only with BSs or only ad-hoc with other MNs or with both. The coverage of BSs is much higher as compared to the transmission range of ad-hoc communication technologies (e.g. if we compare GSM to Bluetooth).

Based on the possible application scenarios, the mobile environment illustrated in Figure 3.1 can be classified into 3 sub-environments:

1. **The mobile infrastructure-based environment** contains only FNs, BSs and a subset of MNs. This subset of MNs can only communicate with each other or with FNs using the services of BSs. Furthermore, the MNs are not able to communicate with each other in an ad-hoc fashion. Figure 3.2 a) shows an illustration of this environment.

2. **The mobile ad-hoc environment** involves only a subset of MNs and mobile sinks of WSNs. These MNs can communicate with each other or with mobile sinks only in an ad-hoc manner, i.e. they cannot communicate with BSs or other FNs. In this environment, mobile sinks construct a gateway for communication with WSNs but only mobile sinks (not SNs) can participate in transactions. Figure 3.2 b) illustrates this environment.

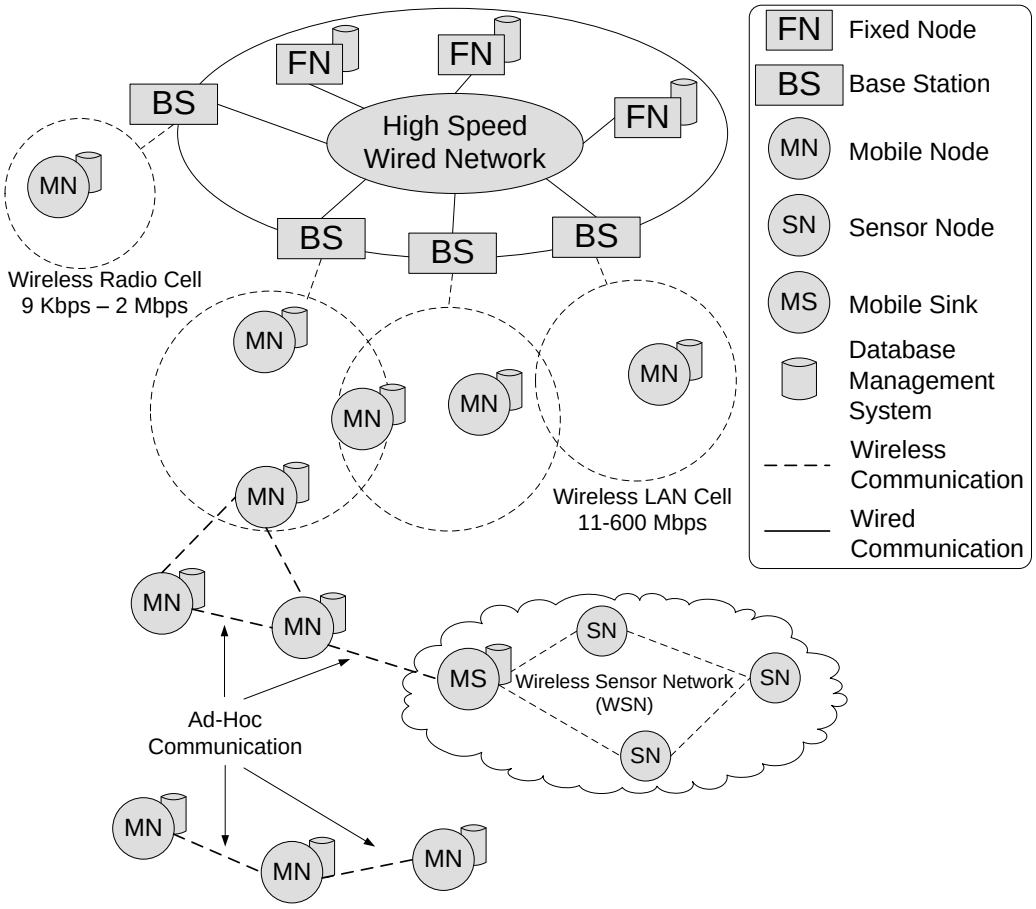

Figure 3.1: Components of the mobile environment

3. **The mobile generic environment** is a combination of both the infrastructure-based and the ad-hoc environments representing a generic mobile distributed model. In this environment, MNs can ad-hoc communicate with each other and also with BSs to reach other MNs or FNs (if infrastructure is available). Figure 3.2 c) shows an illustration of the mobile generic environment.

The nodes usually entail varied hardware and software. In particular, MNs can range from cell phones with restricted storage and processing capabilities to laptops with considerably higher capabilities. For FNs, we do not place any restriction on the computation and storage capabilities. Furthermore, MNs may use different wireless interfaces for communication ranging from low bandwidth and costly links (e.g., GPRS) to high bandwidth and free links (e.g. WLAN). Summarizing, we are dealing with highly *heterogeneous*

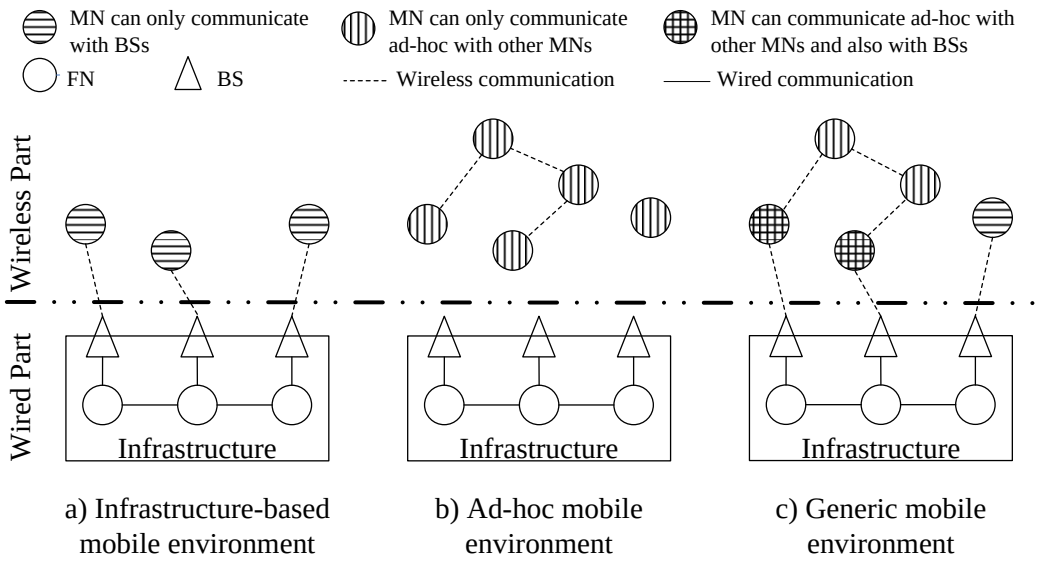

Figure 3.2: Generic vs. infrastructure-based and ad-hoc mobile environments

nodes and links.

We assume that each MN has a Mobile Database Server (MDBS) installed on it, and that a Database Server (DBS) is attached to each FN. Database servers are needed on both fixed and mobile nodes to support basic transaction operations such as Read, Write, Commit and Abort. The nodes are typically equipped with varied *database management systems (DBMS)*. For instance mobile phones employ embedded DBMS such as [Oracle Database Lite], [IBM DB2 Everyplace] while laptops utilize standard DBMS such as [Oracle Database Standard Edition], [IBM DB2].

We consider all distributed database system components (in $M \cup S$) to be *autonomous*, i.e., every component must take the decision to either commit or abort the transaction independently from other components in the network. Components are also able to decide which information to share with the global system and how to manage their local data. The data of the MN may be replicated on a backup database server in the fixed wired network. The synchronization of the data between the MN and its corresponding backup server is done periodically by the user.

In this work, we do not assume any bounds on message transmission times between communicating nodes and also on data processing times on a node.

3.2 Transaction Model

We consider applications, which run on either MNs or FNs and access data located on *mobile* and/or *fixed* nodes. Subsequently, a transaction can originate from any node in $M \cup S$, and the participants in its execution can be any set $P \subseteq M \cup S$. We focus on distributed transactions issued by either MNs or FNs and involving other MNs and/or FNs as participants. A distributed transaction where at least one MN participates in its execution is a *Mobile Transaction (MT)*. Commonly, a MT T_i is defined as a set $F = \{e_{i1}, \ldots, e_{in}\}$ of n "execution fragments" distributed among a set of locations (also sites) in $M \cup S$ [Kumar, 2000; Kumar and Dunham, 1998]. The MN, where T_i is initiated, is termed as *Initiator MN (I-MN)*. The *commit set* consists of all FNs and MNs participating in execution and commit of T_i including the I-MN. FNs and MNs in the commit set are called *participant FNs (P-FNs)* and *participant MNs (P-MNs)* respectively.

Transaction commit protocols are generally based on the existence of at least one *coordinator (CO)*, which is responsible for coordinating the execution of the corresponding transaction. For different transaction and mobile system models, different nodes may play the CO role which requires special capabilities such as stable storage. The CO is responsible for storing information concerning the state of the transaction execution. Based on the information collected from the participants of the transaction, the CO takes the decision to commit or abort the transaction and informs all participants about its decision.

3.3 Perturbation Model

Designing perturbation-resilient transaction commit protocols essentially requires the identification of relevant perturbations, i.e., operational/environmental constraints and failure modes that can occur in the considered environment and disturb commit functionality. The following sections classify and enumerate these aspects.

3.3.1 Classification of Perturbations

We recognize the following two main classes of perturbations: environmental/operational constraints and failures (Figure 3.3). We classify the environmental constraints relevant to commit protocols into heterogeneity (of nodes and links), unstable storage and energy constraints. Failures of the mobile environment are classified into communication and node failures. Communi-

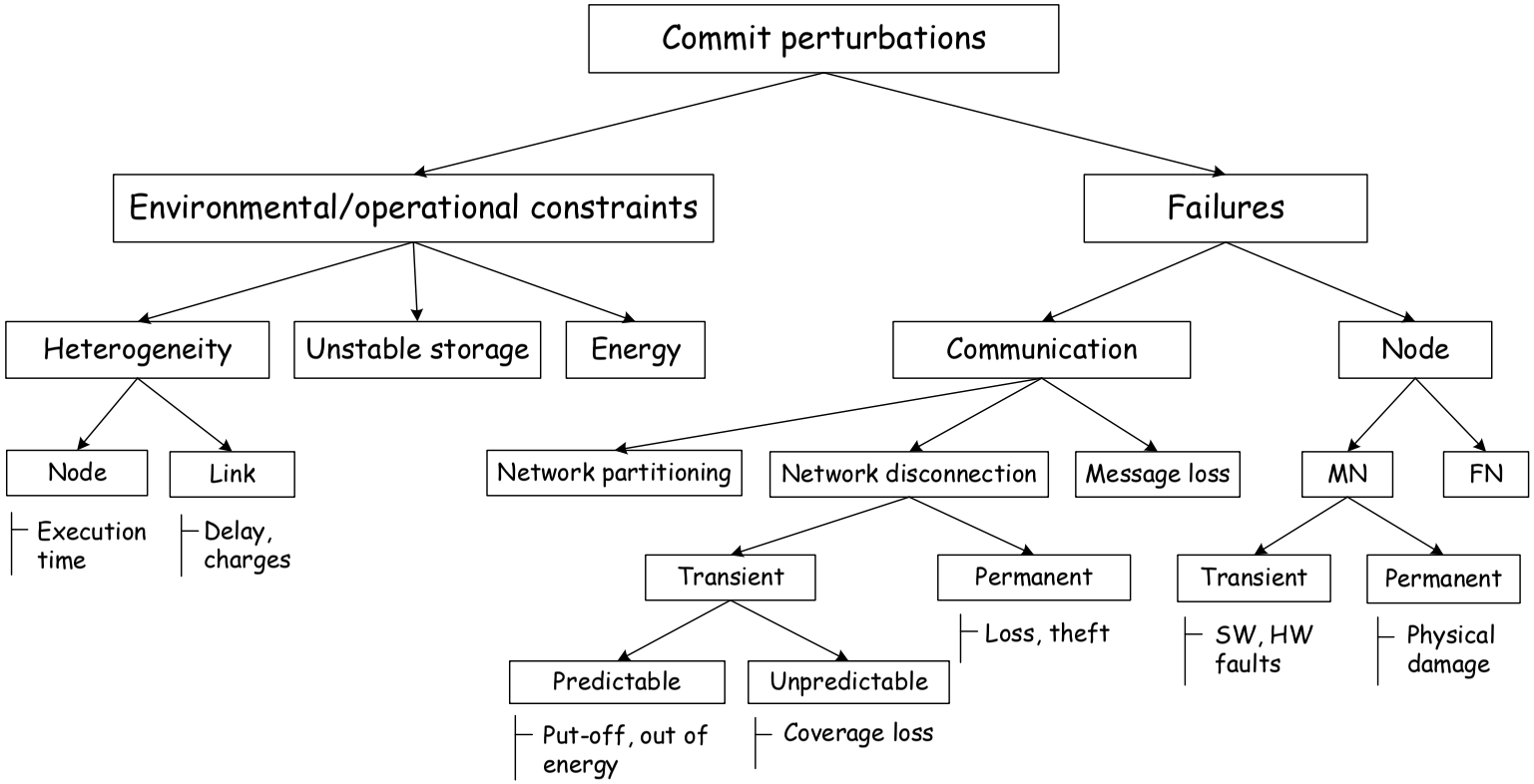

Figure 3.3: Classification of perturbations

cation failures are due to message loss, network disconnection and network partitioning. We divide network disconnection into transient predictable, transient unpredictable and permanent. Examples of transient predictable network disconnection include the class of planned disconnections (such as put-off or reboot) and situations in which the network coverage degradation is predictable such as when the wireless signal is continuously getting weaker. Another situation in which the MN can predict its disconnection is when the battery charge is low (disconnection due to node failure). Network partitioning occurs in general only in mobile ad-hoc or generic environments and splits the network into partitions that are not able to communicate between each other. Node failures are either MN or FN failures. MN failures are in turn either transient or permanent.

The comprehensive classification of perturbations in mobile environments helps simplify the identification of the main building blocks to guarantee perturbation-resilience for common mobile system classes. These common mobile system classes with their corresponding building blocks build the basis for the protocol family that we will present in Chapters 5, 6 and 7.

3.3.2 Operational/Environmental Constraints

The considered mobile environment is constrained mainly by the characteristics of *MNs* and *wireless links*. MNs (ranging from laptops, personal digital assistants (PDAs), to cell phones) intuitively possess less computational resources than FNs, for instance processor speed and storage capacity. Especially, some MNs possess limited memory space which restricts the amount of data storable on them. These resource constraints increase the time MNs need to execute transaction fragments or may even lead to execution failures. Furthermore, MNs have no stable storage since they are carried by users, incur operational wear and tear and can also be easily lost or stolen. Additionally, data replication strategies typically used in mobile environments have limited capabilities. Most replication strategies proposed in the literature such as [Pradhan et al., 1996] rely on BSs to replicate data of the MNs. BSs are not always available (mobile ad-hoc environment) and belong to a third party service provider and it is not clear whether these providers want to contribute to achieve such goals or not. Another issue is the cost of storing the data on these BSs. Due to these issues the memory storage on a MN cannot be considered as stable.

MNs rely on finite amount of *energy* provided e.g. by batteries, which implies that they can run out of energy anytime thus losing information about the status of execution stored on their volatile storages. Two of the most important sources of power consumption are transmissions and memory

accesses. We note that, for a MN, transmitting data consumes around three times as much energy as receiving the same amount of data [Forman and Zahorjan, 1994].

Wireless network characteristics also change more frequently than those of wired links. For example, the effective available bandwidth is highly dynamic. This depends on the wireless technology (GSM, UMTS, WLAN, satellite, ...), access coverage, and number of MNs that have to share the wireless medium. Other key characteristics of the wireless links are latency and communication costs. These characteristics lead to considerably varied reliability/availability and connectivity of MNs.

The limitations and characteristics listed above outline the variety of operational/environmental constraints for the mobile environment being different from those in fixed wired networks. These operational/environmental constraints also complicate the design of efficient commit protocols.

3.3.3 Failure Modes

We now outline the considered failure modes classified into the primary classes of communication and node failures.

Communication Failures

These constitute the majority of failures in the mobile environment. We distinguish between two types of communication failures:

Message Loss: Message exchange between the MN and the BS or between the MNs (in ad-hoc mode) is highly vulnerable to loss due to the high bit error rate of wireless links, and also because of network congestion and collisions. Message loss is much more probable to occur in mobile environments than fixed ones and needs to be explicitly taken into consideration in the design of mobile systems.

Network Disconnection (or Link Disruption): Given its mobile nature, a MN can enter a geographical area out of coverage of any BS so that it loses its connection to the network. The MN is said to be *disconnected* from the rest of the network. While disconnected from the network, the MN is not able to send or receive messages. As network disconnection [Kistler and Satyanarayanan, 1992] is a common occurrence in mobile scenarios, it needs to be explicitly considered in the system design. Network disconnections can be either transient or permanent.

Network Partitioning: Network partitioning occurs only in mobile ad-hoc or generic environments. Due to the inherent node mobility and autonomicity, the mobile ad-hoc environment can easily get partitioned and reconnected. As shown in [Hähner et al., 2004, 2007], network partitioning is frequent and unpredictable in mobile ad-hoc networks. As network partitioning is the norm rather than the exception in a mobile ad-hoc environment, it needs to be explicitly considered in the design of atomic commit protocols.

Node Failures

We distinguish between MN and FN failures. For MNs, we identify two main failure classes, i.e., *transient* and *permanent* failures. We do not consider malicious failures such as Byzantine faults or intrusions, but in future work we want to extend the fault model incorporating malicious faults.

Transient MN Failures: These occur mainly due to either software or hardware faults and usually disappear if the MN reboots. A further cause of transient failures is the lack of battery power to sustain operation of the mobile device. Transient failures are the most probable failures of MNs in the mobile environment. Opposite to network disconnection or partitioning, in the case of a transient MN failure the content of the volatile storage of the MN and consequently the state of its recent computations is lost.

Permanent MN Failures: These are irreparable failures such as loss, theft or physical damage of the MN itself or its non-volatile storage where the data and logs are stored (media failure). Consequently, all the data stored in the MN is lost. Despite the fact that permanent MN failures are rather rare in mobile environments when compared to transient MN failures, this failure mode of MNs does occur because of the mobility and size of MNs.

FN Failures: We assume that if a FN crashes, then it stops receiving, sending and processing messages until it recovers after a finite but unbounded amount of time. This is known as the crash-recovery model.

3.4 Chapter Summary

We presented in this chapter the different models our work is based on. We proceed now by describing the perturbation-resilient atomic commit framework in the next chapter where all the techniques will be described to cope

with the environmental constraints and failure modes identified and detailed in this chapter.

Chapter 4

Perturbation-Resilient Atomic Commit Framework

To cope with different mobile environments, we present in this chapter our perturbation-resilient atomic commit framework which builds the basis for the modular framework of transaction atomic commit protocols described in the next Chapters 5, 6 and 7. First, we introduce a set of application scenarios that show the need to investigate the atomic commit problem for the different types of mobile environments (defined in Section 3.1), i.e., the mobile infrastructure-based environment, the mobile ad-hoc environment and the mobile generic environment. Next, we provide the design requirements for developing atomic commit protocols in mobile environments in general along with our identified performance requirements. Finally, we present the perturbation-tolerance design techniques and mechanisms relevant to each perturbation class identified in Section 3.3. The choice and investigation of these perturbation-tolerance design techniques and mechanisms are driven by the performance and design requirements defined in this chapter.

4.1 Application Scenarios

Due to the growth of mobile devices in computational power and storage capacity, new applications are emerging where strict data consistency and consequently strict transaction atomicity is required. In the following some application scenario classes are presented to highlight the strict atomicity requirements by applications.

4.1.1 Bank/Stock Transactions

This type of application scenario includes mobile commerce (m-commerce) applications where users can buy or sell goods using their mobile devices while involving bank servers in fixed networks to accomplish their transactions. These applications usually run in mobile infrastructure-based environments and increasingly in mobile generic environments involving WSNs for checking the availability of goods. The role of WSNs in this scenario is to keep track on availability of certain type of goods and their corresponding quantity. In this case, transactions can be processed automatically without human intervention but using an on-demand availability check. A sell transaction in this case can involve different people located in different places in the mobile market and using mobile devices equipped with different wireless interfaces. Strict atomicity guarantees in this scenario that the buyer gets the items and services he paid for and that the seller is not reserving his goods for potential buyers for long periods of time waiting for the buyer to pay for these goods. This might happen if strict atomicity is not supported and the buyer declares his interest in buying some items and never appears to complete the initiated transaction.

4.1.2 Coordination across Autonomous Vehicle Systems

Mobile transactions are needed in such type of application scenarios for the purpose of coordination for safe navigation of unmanned autonomous networked vehicles. These vehicles are also equipped with different wireless interfaces and can communicate directly in an ad-hoc manner or using the wired infrastructure if available and accessible. Such type of application can run in mobile ad-hoc or mobile generic environments. Transactions are needed for coordination between unmanned vehicles at traffic intersections in order to consistently update the databases of these vehicles (and possibly authority databases) with the passage order of the present vehicles at the intersection (Figure 4.1). These vehicles need to agree on an order how

they will pass the crossing. Prior to their actual passing, this order information needs to be agreed upon and recorded atomically to their corresponding databases. This stored data might be needed by insurance companies or the police department in case an accident occurs. Intelligent transportation systems and platooning systems [Swaroop, 1994; Swaroop and Hedrick, 1996] are further possible scenarios where mobile transactions might be needed.

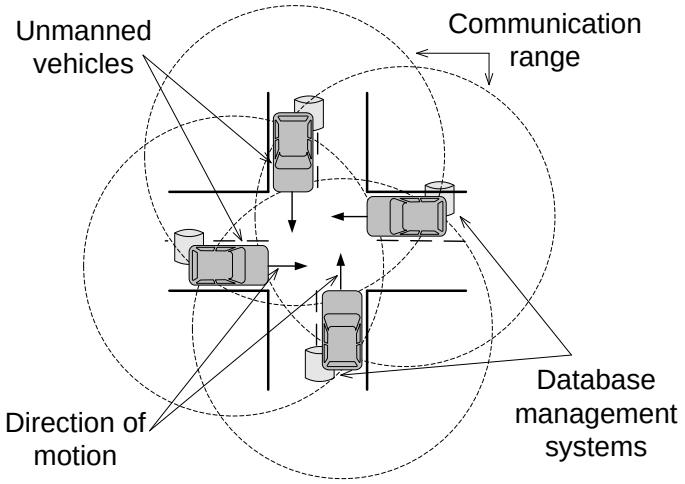

Figure 4.1: Coordination across autonomous vehicles (livelock scenario)

Other scenarios for mobile ad-hoc environments might include additionally to coordination payment and mobile commerce services. A detailed description of a commercial mobile ad-hoc application which represents a radio dispatch system between taxis can be found in [Huang et al., 2005]. The radio dispatch system is described as a novel and plausibly realistic application scenario for mobile ad-hoc environments. The proposed system is then evaluated in [Huang et al., 2005] from both financial and technical perspectives to provide a complete picture of its feasibility.

4.1.3 Mobile Gaming

This is a potentially interesting and promising application of mobile transactions in mobile ad-hoc (and possibly generic) environments. We consider mobile role-based games where players can either exchange or buy/sell virtual characters with virtual items or real money. Game information is stored on local databases (and if accessible on central databases). If a player's action affects one or more other databases, mobile transactions are required to guarantee the consistency of the game data.

4.1.4 Disaster Management

Mobile devices can be used in case of forest fires to supply fire fighters with their current position data, burning densities and to coordinate their actions, e.g. rescue actions [Obermeier et al., 2009]. Atomic decisions are required in this type of scenarios to safely coordinate the next action of the participating fire fighters where the agreement of the involved fire fighters is crucial for their security in such type of missions. Such type of application scenarios run in general in mobile ad-hoc environments because of the lack of infrastructure in disaster scenarios in general.

4.1.5 Health-Care Ambient Intelligence

Health-care ambient intelligence [Aarts et al., 2001] scenarios provide an example of mobile generic environment scenarios as defined in our system model in Section 3.1. Mobile transactions can help in such type of scenarios to keep data about monitored old people living alone consistent among family members and other institutions such as hospital and insurance for example.

4.2 Design Requirements and Issues

We now present the design requirements of resilient atomic commit protocols for mobile transactions. A basic question is the need for new design requirements for atomic commit protocols in mobile environments. The difference between mobile and fixed environments is that perturbations in mobile environments are not the exception but often the normal case. Thus, we need to define the boundaries of our framework in terms of design requirements. We identify the following main requirements and design issues to be handled by the framework.

4.2.1 Perturbation-tolerance and recovery

To build resilient atomic commit protocols, it is essential to define a comprehensive categorization of perturbations and a set of techniques to cope with environmental constraints and recover from failures. The categorization of perturbations assists the protocol designer in identifying the main concerns and developing appropriate solutions. The overall objective for perturbation-tolerance is to maximize the commit rate. A naive approach to provide for fault-tolerance is to abort the MT each time a failure occurs and to restart it (e.g., after a back-off time or after the failure disappears). This simplistic approach introduces a large overhead for the successful participants (due

to frequent re-execution of the fragments) and requires some external intelligence (either from the user or from the ability of the system to detect failures). Therefore, we introduce the delay-tolerance design requirement for MT.

4.2.2 Delay-tolerance/-awareness

Masking latent faults such as long disconnections imposes that the MT execution time can be delayed till local "Commit"/"Abort" decisions can be collected. This implies that a MT can last for minutes or even hours. Thus, the interest is on developing *delay-tolerant or delay-aware transactions* where users can sacrifice latency for atomicity. For example, international bank transactions can last days due to heterogeneous processing and regulatory issues across the countries. We can easily expect that the application/user is able to specify an appropriate (tolerable) *lifetime* for each initiated MT.

4.2.3 Efficiency

The efficiency of commit protocols is measured in terms of exchanged messages and resource blocking time. The classical approach to improve the efficiency of such protocols is to reduce the communication overhead (number and size of messages) and to minimize the resource blocking time. The rationale behind minimizing resource blocking time is that transactions, especially those executing on FNs, often lock expensive resources. Transactions are isolated from each other by locking all relevant data needed by them. As long as the locks are held, no other transaction can access the data, i.e., data or resources are *blocked*. The more transactions per time unit an application can process, the better the system's scalability and throughput. If resources are blocked, transactions using them are delayed waiting for the resources to be unlocked resulting in reduced throughput. For this reason the blocking time, especially of FN resources as they are frequently much more loaded than MNs, should be minimized.

4.2.4 Scalability

Commit protocols are considered to be scalable if they can support a growing number of participants without sacrificing efficiency. Resource blocking time as well as capabilities of the CO to handle more transactions per time unit are the main factors that determine the scalability of commit protocols.

The efficiency and scalability design requirements are orthogonal to the delay-tolerance requirement and imply a key challenge for the generalized

commit framework that we target.

4.3 Methodology

Our primarily goal in this chapter is to offer perturbation-resilience optimizing the performance of commit protocols in mobile environments. This chapter focuses on transparent perturbation-tolerance techniques which do not require any intervention from the user and mask the perturbations cited in our perturbation model (Section 3.3). We also discuss different techniques to mask the environmental constraints in order to optimize the overall system performance and resilience.

The perturbations were classified in Section 3.3 into a set of discrete classes to help simplify the description of the solutions by supporting a modular and hierarchical approach. In this chapter, we present perturbation-tolerance design techniques relevant to each class driven by performance optimization requirements defined in Section 4.2. As several contemporary efforts for commit protocols in mobile environments exist, we review the existing strategies and maximize their reuse. Especially, we aim at reusing the results of the mature work existing for fixed networks.

We proceed progressively by tackling every identified perturbation in Section 3.3 and propose for every identified perturbation class a set of perturbation-tolerance mechanisms and techniques that will be used in the next Chapters 5, 6 and 7 to build a modular framework of transaction atomic commit protocols for mobile infrastructure-based, ad-hoc and generic environments, respectively. We start by investigating mechanisms and techniques used to mask **environmental constraints**.

We proceed then by investigating perturbation-tolerance techniques for **network disconnections**, **message losses**, **node failures**, and finally **network partitioning**.

4.4 Coping with the Environmental Constraints

We now discuss the main techniques to provide resilience to the identified environmental constraints, i.e., node/link heterogeneity, unstable storage and energy.

4.4.1 Heterogeneity of Nodes and Links

We start from a scenario where only homogeneous fixed nodes communicate through high speed networks of homogeneous links using standard commit protocols such as 2PC or 3PC. Finding a timeout (TO) value after which the CO can abort the transaction if it does not receive the votes of all participants is straightforward since the costs involved with restarting the transaction are not high. This timeout value should only mask the slightly oscillating execution times and communication delays due to varying node and network loads. These timeout values do not usually exceed the range of some seconds.

Next we consider the same scenario but with heterogeneous nodes, i.e., the participants can have different capabilities for CPU processing, memory, etc. Finding an appropriate timeout value now becomes challenging since the time needed to execute different fragments of a transaction can vary considerably from one participating node to another. As restarting the transaction with a higher timeout value is not costly in the fixed environment, this would be a suitable decision. The real challenge is when some of the heterogeneous devices participating in the transaction become mobile and use not only high speed networks but also, like in our system model, various wireless communication links. So setting an appropriate timeout value by the CO in this new scenario can reduce the costs of restarting the transaction. Also modifying the commit protocol to use less messages can further reduce these costs and save the limited bandwidth.

In [Kumar et al., 2002] the authors presented a timeout based approach to deal with node and link heterogeneity. Each P-MN computes an *execution timeout (E_t)* - an estimated upper bound for the time to complete the execution of its transaction fragment - and a *shipping timeout (S_t)* - an estimated upper bound for the time to compose updates and to send them to the CO. The P-MN sends both timeouts to the CO. Both timeouts have to account for the environmental constraints related to the P-MN and the wireless link it uses. These timeouts can be extended if needed. The CO of the MT sets its timeout according to the time needed by the participants to execute their fragments and to send their votes to either commit or abort the transaction. Details on how this timeout is set are not provided in [Kumar et al., 2002]. However, this approach will always abort the MT if one of the participants is considerably slower than the initiator as illustrated in Figure 4.2. This figure shows that in case one of the participants is considerably slower than the initiator, the timeouts of the slow participant are received after the expiration of the CO timeout defined in this case by the initiator, i.e., after the abortion of the MT. Therefore, we extend this approach to an advanced timeout handling as follows.

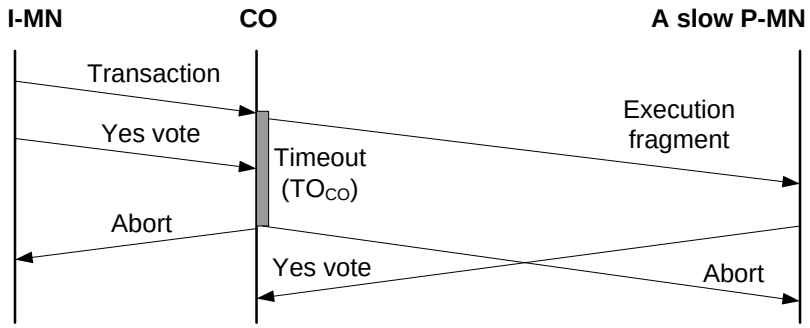

Figure 4.2: Timeout selection in a heterogeneous scenario

We let the I-MN inform the CO about the estimated/desired *lifetime* of the MT. The lifetime of a transaction provides the maximal timeout the CO should wait to take the decision about the outcome of the MT. It takes into consideration the environmental constraints including heterogeneity of MNs and wireless links. The lifetime can either be set by the application or estimated by the initiator or CO based on previous experiences and observations. The CO requests from all P-MNs their estimated timeouts $TO_{P\text{-}MN_i}$. If the CO receives a lifetime from the I-MN, it sets its own timeout $TO_{CO} = lifetime$. If the lifetime is undefined for any reason, the CO updates its own timeout every time it receives a timeout estimation from one P-MN by setting it to the maximum of all received $TO_{P\text{-}MN_i}$ ($TO_{CO} = \max\{TO_{P\text{-}MN_i}\}$). If TO_{CO} expires before receiving a timeout from one P-MN, the CO aborts the MT. In this case the CO additionally needs to identify the slowest P-MN and to estimate the time needed to receive its timeouts (Figure 4.2). We present a detailed discussion of the scenario where the lifetime is undefined in Section 4.5 below.

As described above, for heterogeneity of nodes and links, an enhanced timeout handling is proposed. The two remaining environmental constraints, i.e. unstable storage and energy impact mainly the selection of the CO. Therefore, unstable storage and energy will be used in the following to justify the choice of the CO as a FN in case such a node is available in the studied scenario, otherwise other strategies such as multiple MNs playing the CO role should be considered.

4.4.2 Unstable Storage

We consider the environmental constraint of unstable storage of MNs and derive its impact on the design of commit protocols. As mentioned before in Section 3.3, the storage of MNs is usually not stable making them even less

suitable for a CO role since it is a key element of atomic commit protocols. Due to unstable storage, if a single MN plays the role of a CO, it can lose all the information related to a transaction and therefore it will not be able to take an Abort or Commit decision. This is similar to a permanent crash of the CO. Furthermore, if the CO runs out of memory and is not able to store required information about the state of the transaction, it has to abort the transaction even after receiving "Yes" votes from all participants.

4.4.3 Energy

The CO sends and receives relatively high number of messages (compared to other participants) in order to take a decision on the outcome of the transaction and also to ensure that every participant is informed about this outcome. Sending and receiving this relatively high number of messages consumes a lot of energy on energy-limited mobile devices making them less suitable for CO role.

The unstable storage and energy constraints make a single MN unsuitable for playing the CO role. The CO role requires a powerful node in terms of computation and memory capabilities having a stable storage and not relying on a finite energy source. This substantiates the selection of a FN to play the CO role whenever possible. If a FN is not available in the investigated environment, multiple MNs should be considered to play the CO role as will be discussed in Section 4.8.

Due to the focus on failure resilient commit, we now consider failures in detail in the following sections covering *(1)* network disconnection, *(2)* message loss, *(3)* node failures and *(4)* network partitioning.

4.5 Tolerating Network Disconnections

Traditional commit protocols trigger Abort if at initiation one or more participants are not connected. These protocols tolerate network disconnection durations of some seconds (using timeouts) and trigger Abort if no connection is observed during this short timeout. These solutions are typically not suitable for mobile environments since the probability that all (mobile) participants are connected at initiation time is usually low. This can be explained by the fact that disconnection time of MNs can range from some minutes to hours or days, or even permanently due to physical damage, theft or loss. In the following, we provide design principles for tolerating three specific classes of network disconnection, i.e., transient predictable, transient unpredictable and permanent (Figure 3.3).

A possible solution to mitigate network disconnection has been proposed by Unilateral Commit for Mobile (*UCM*) [Bobineau et al., 2000]. UCM does not comprehensively address disconnections and proposes a one-phase protocol where the voting phase of 2PC is eliminated. The CO acts as a dictator imposing its decision on all participants. Thus, UCM assumes strong assumptions on the execution of fragments, integrity constraint check, serialization and logging on the P-MNs and P-FNs. These assumptions restrict and violate the autonomy of the entities participating in transaction execution. Also these assumptions are usually hard to fulfill in mobile environments because they require a fully connected network, which is not common in mobile networks.

As a generic solution for these classes, we suggest creating representatives/proxies of MNs in the fixed part of the network. Introducing representatives is inspired by the M-2PC protocol [Nouali et al., 2005] and the work presented in [Serrano-Alvarado et al., 2003]. In M-2PC the representatives take part in the transaction execution when P-MNs delegate their commitment duties to them. We extend the role of these representatives which can act on behalf of MNs from the beginning and mask their disconnections. This is similar to a lightweight replication of commit data and commit state of the MN in the fixed part of the network, i.e, these representatives will store required information concerning the state of execution of the MT and also the message traffic from and to the corresponding P-MN to be able to act on its behalf in case perturbations occur. These representatives are henceforth termed as mobile node agents (MN-Ag). MN-Ags are implemented on FNs and are provided as a service to the user by its service provider (the service provider provides infrastructure-based communication facilities to the P-MNs). We consider a MN-Ag per MN and per all the transactions involving the MN. The MN-Ag can play the CO role if the corresponding P-MN is the initiator (I-MN) of the transaction. Figure 4.3 shows a scenario where the MN-Ags act on behalf of their corresponding P-MNs in case of network disconnection. In this scenario the MN-Ags buffer the messages received from the CO and forward them to the corresponding P-MNs when they reconnect.

Besides disturbing the Commit/Abort decision process, network disconnection substantially impacts the *blocking time* of FN resources. Using a classical transaction commit protocol such as 2PC, disconnections of P-MNs can block the valuable resources of FNs for an intolerable long time period. Thus, it is crucial to minimize this blocking period. We suggest decoupling the commit of P-MNs from that of P-FNs. The transaction execution is consequently split into *two phases* (Figure 4.4). We call the first phase *pre-commit phase* where "sufficient" information is collected from P-MNs after finishing the execution of their corresponding fragments. The second phase

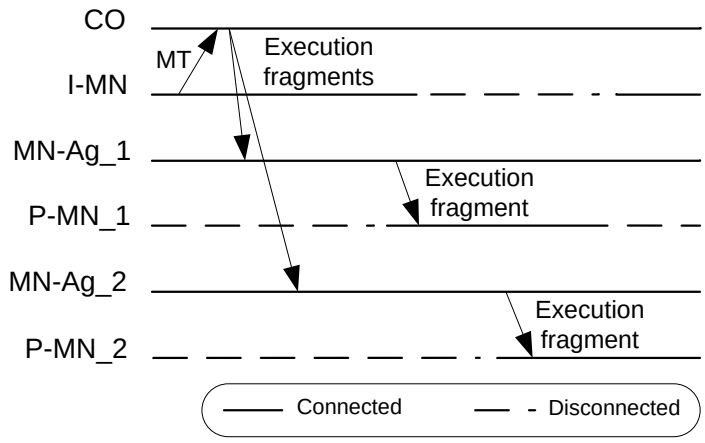

Figure 4.3: Tolerating network disconnection – Agent concept

called *core phase* involves only P-FNs. Therefore any classical transaction commit protocol established in fixed networks such as 2PC or 3PC can be used. If the first phase fails, i.e., in case the CO receives at least one "No" vote or TO_{CO} expires before the CO receives all votes, then it is useless to progress with the second phase and consequently the blocking of P-FNs' resources is avoided. Otherwise, the blocking time of resources on P-FNs is determined by the blocking time of the core phase protocol.

Therefore, the decoupling allows preventing network disconnection, mainly caused by P-MNs, from affecting P-FNs. For the different network disconnection types different precautions to design the pre-commit phase are discussed in the following.

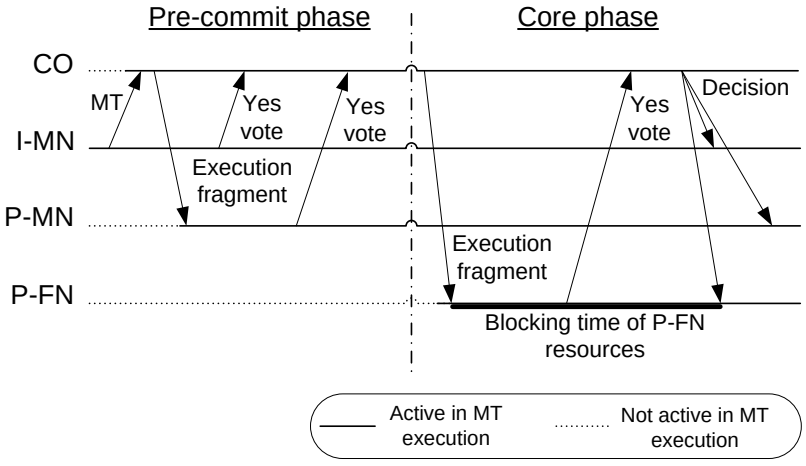

Figure 4.4: Tolerating network disconnection – Decoupling concept

4.5.1 Transient Predictable Network Disconnections

For the considered classes of network disconnection, we assume that only the initiator is connected at the time when the transaction is issued. Other P-MNs are not required to be connected. A transient predictable network disconnection allows the node to exchange some messages with other entities participating in the execution or coordination of the transaction just before it becomes disconnected. Thus, the node can likely inform other nodes about its predicted disconnection (Figure 4.5). One solution to this problem could be to wait (the CO) for a predefined amount of time (TO_{CO}) and abort the transaction. This solution increases the number of transaction Aborts in this environment and also the costs associated with the re-initiation of aborted transactions in term of messages and energy. The aborted transaction should be also delayed to be executed in the near future which can affect other dependent/related transactions.

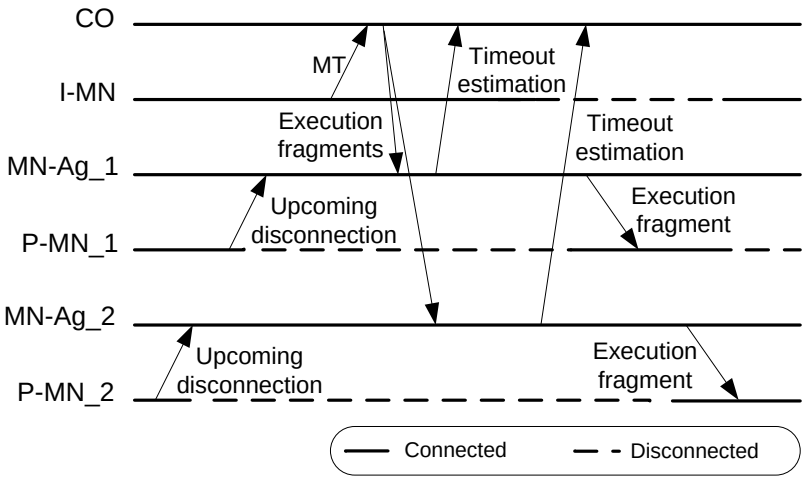

Figure 4.5: Tolerating predictable network disconnection

For transient predictable network disconnection the P-MN knows approximately when it will disconnect. If it additionally knows how long it will disconnect, then a timeout extension ($TO_{ext}(P\text{-}MN)$) is easy to determine for the P-MN. $TO_{ext}(P\text{-}MN)$ is an update of $TO_{P\text{-}MN}$ which should be sent to the CO to update its timeout TO_{CO}. If the P-MN lacks this information, a table of possible scenarios for network disconnection and the corresponding estimated disconnection times can be used. As a representant of the P-MN, the MN-Ag can take decisions about $TO_{ext}(P\text{-}MN)$ on the P-MN's behalf when the P-MN is disconnected (Figure 4.5). This decision should be taken in case of transient predictable network disconnection based on the

information sent by the corresponding P-MN before disconnecting.

For delay-tolerant transactions $TO_{ext}(P\text{-}MN)$ can be evaluated based on the P-MN's behavior in the past. A history of previous disconnections can assist in estimating an appropriate value for the $TO_{ext}(P\text{-}MN)$ for future disconnections.

4.5.2 Transient Unpredictable Network Disconnections

This type of network disconnection refers to the case when the P-MN disconnects; however no other entity participating in the execution of the transaction is updated on the state of the P-MN disconnection. This occurs as the P-MN was not able to communicate with its MN-Ag or with any other participant to inform about its actual status before disconnection. When a transient unpredictable network disconnection occurs, a timeout extension can only be triggered by either the CO or the MN-Ag, where the timeout selection may be less suitable. For this we suggest that each P-MN specifies for its MN-Ag a *default timeout extension value* for the case the network disconnection is unpredictable. The MN-Ag should also have the possibility to extend the timeout of its corresponding P-MN when needed. We suggest here that the MN-Ag develops an experience log for transient unpredictable network disconnection tailored to its corresponding P-MN. For example, to first estimate the disconnection cases of short duration (tunnel (GSM), handover (GSM-UMTS-WLAN), software (SW) transient failures) and subsequently of medium length (hardware transient failures) and for long-duration reasons (e.g., discharged battery) and in worst case long-duration network disconnection may become permanent network disconnection due to damage or loss.

4.5.3 Permanent Network Disconnections

When the P-MN is lost, stolen or damaged, we consider that permanent network disconnection has occurred. In this case the transaction should be aborted. If the P-MN has voted with "Yes" before crashing permanently the transaction could be committed if the changes done by the P-MN on his local database are available to its MN-Ag or to the CO. These changes can be propagated to the main copy of the data located e.g., on a backup server on the wired network or to the user. In this case, the MN-Ag can be used to store this information and will be responsible for its propagation to the backup server or to the user whenever needed.

4.6 Tolerating Message Losses

Message loss is a common occurrence over wireless links. A message loss can be tolerated by using *acknowledgments (Acks)* and *timers* with appropriate timeout values. Acks constitute an overhead in term of messages for the considered mobile environment. Therefore, the number of Acks exchanged during the execution of atomic commit protocols should be kept minimal. Timers can also be used to detect loss of messages by setting an appropriate timeout value after which the message is assumed to be lost. Trade-off can be obtained when to use Acks or timers or a combination of both.

4.7 Tolerating Node Failures

As mentioned before, we classify node failures into MN and FN failures. We note that a node failure implies network disconnection besides data loss, which we investigate in the following subsections. MN failures can either be transient or permanent. To cope with transient failures, local recovery is used. Permanent failures can be tolerated using replication.

4.7.1 Transient Mobile Node Failures

Local recovery is the set of operations that must be performed locally by a node after a transient failure to recover to a correct, consistent and failure-free state. For this purpose, a set of precautions has to be conducted during the failure-free operations. To identify these precautions, it is needed to categorize the situations where recovery is needed. For example, if failures occur they should not result in loss of commit data or commit state on a node or a subset of nodes. We classify these situations into isolated node failures and combined node failures.

Isolated node failures consist of scenarios where only one P-MN is affected by a transient failure (e.g. battery depletion) during the execution of a MT. All these failures result in the loss or corruption of the content of volatile storage. One solution is logging of all needed operations related to the execution of the transaction commit protocol on *a stable storage*. When a failure occurs the recovery is performed based on the logged information. But which data should we log? How frequent should logging be performed? And where to log this data if the MNs are not assumed to have a stable storage? Answering these questions requires a detailed knowledge about the transaction commit protocol and the interaction of the participants. We will detail this issue while discussing transaction atomic commit solutions in

mobile infrastructure-based environments in Chapter 5.

Combined node failures include scenarios when more than one node at a time observe a transient failure and these failures can be either similar or different in nature. In these situations recovery may need global information about the state of execution of the transaction depending on the transaction commit protocol that defines which information needs to be exchanged between the different nodes either participating in the transaction or responsible for its coordination. Usually, this global information is stored and managed by the CO.

4.7.2 Permanent Mobile Node Failures

This type of failure can be tolerated if the data or the logs related to the execution of MTs on MNs are replicated (e.g., on MN-Ags) before committing the transaction. Logs should not be replicated if the transaction is aborted since the MT should not have any effect on the data stored on the MN in this case. If the final decision is Abort, the logs are only needed locally.

4.7.3 Fixed Node Failures

Decoupling, as discussed in Section 4.5, allows for easy reuse of existing techniques to handle FN failures. For the sake of completeness, we provide a short overview of the basic fault-tolerance recovery mechanisms. For a detailed survey we refer the reader to [Elnozahy et al., 2002]. An important fault-tolerance mechanism useful to tolerate FN failures is *checkpointing*. A checkpoint is a record of a consistent state that existed on a node at some time in the past. Checkpointing is usually used along with *logging*. A log represents a durable record or history of the significant events such as Write, Commit and Abort that have occurred at this site either since the start or since the last checkpoint. If a failure occurs the node needs to rollback to a consistent state. Recovery strategies can be divided into roll-backward recovery, roll-forward recovery or a combination of both. *Roll-backward recovery* brings the system back to a previous correct and consistent state. Checkpoints are made periodically during normal operations by recording (on stable storage) the current state. This represents a big performance overhead for on-going transactions. After a failure the state can be restored from the checkpoint info. *Roll-forward recovery* brings the system to a new correct state after a crash. This may involve asking another site what the current state is if data is replicated. *The combination of both* (often used in databases) uses both checkpoint and recovery logs and proceed as follows:

(a) Take a checkpoint and delete old log and start new log, *(b)* log all significant messages, transactions, etc, up to the next checkpoint, and *(c)* when recovering from failure, restore checkpoint state then replay log and re-do these operations.

If the CO is a FN, we adopt the fault-tolerance and recovery strategies discussed above for FN failures. In addition we investigate which information related to the execution and coordination of the MT should be stored on a stable storage to allow the CO to recover from node failures. Beyond storing the information related to the MT such as the commit set, the corresponding execution fragments and the identity of the CO, the CO needs to maintain information about the status of execution of every execution fragment. We distinguish between the following states: (a) *Idle*, (b) *active*, (c) *pre-committed* (only for MNs and if decoupling is used), (d) *committed* and (e) *aborted*. Idle means that the participant has not started executing its fragment yet. After starting the execution the state becomes "active". If decoupling is implemented and its first phase succeeds, the state of P-MN is updated to "pre-committed", otherwise the state is "aborted". The state "committed" is reached when the whole MT is committed, otherwise the final state is set to "aborted".

4.8 Tolerating Network Partitioning

Mobile ad-hoc environments are characterized by the lack of infrastructure and also by the self-organization of the network. Ad-hoc scenarios show frequent and unpredictable network partitioning as mentioned in Section 3.3. MNs in such scenarios are the only participants in the execution of MTs. As MNs do not connect to any infrastructure, the CO of the MT is required to be a MN. A MN is not assumed to have stable storage and is therefore not able to play the CO role alone unless specific assumptions on the capabilities of such a MN can be made (however, same capabilities as a FN are in general not realistic). Failures of the single mobile CO usually lead to the blocking of all participants. Two extreme cases are possible: Only one CO is defined by introducing a more powerful MN (with additional assumptions on it such as stable storage) or every single participant in the MT is CO. We believe that only a subset of the participants should play the CO role as justified later in this section.

The lifetime concept for infrastructure-based scenarios introduced in Section 4.4 can also be used in ad-hoc scenarios. Estimating the appropriate lifetime value depends on multiple factors. A key issue is the network connectivity, which primarily depends on mobility parameters such as speed of

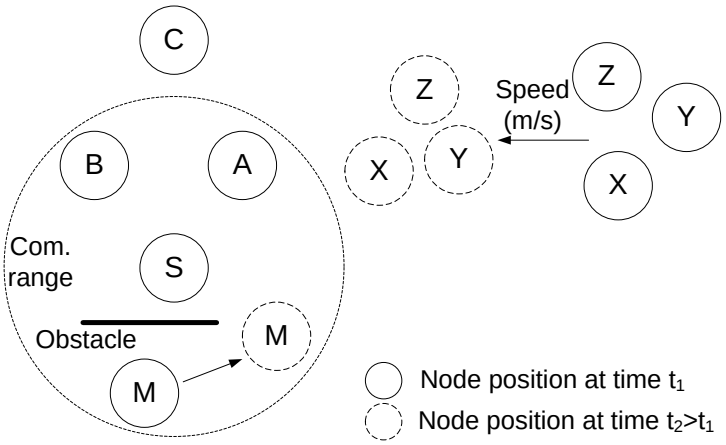

Figure 4.6: Factors for estimating MT lifetime in ad-hoc scenarios

MNs, and their communication parameters as depicted in Figure 4.6. These variables make estimating lifetime in ad-hoc scenarios a challenge. Applications initiating delay-aware mobile transactions should be at least able to compute how long they will be able to wait before receiving the results of the initiated MT. This time can be used as the lifetime of the initiated mobile transaction or can be adapted to the current state of the underlined mobile ad-hoc network. In this work, we do not assume synchronized clocks across the mobile entities. Thus the lifetime can elapse on different times for different participant MNs. This issue should be considered while designing an appropriate MT solution.

Given frequent network partitioning, it is challenging for ad-hoc scenarios to disseminate the fragments of the MT to their corresponding MNs. For this, partition-aware dissemination protocols can be used such as Hypergossiping [Khelil et al., 2007].

As described in Chapter 2, [Bose et al., 2005; Böttcher et al., 2007] propose the use of a *cluster of coordinators* preferably in single-hop distance from each other to avoid blocking of mobile participants in case one CO fails. The cluster of COs elects a single *main coordinator* and uses the 3PC protocol [Skeen and Stonebraker, 1983] to agree on a consistent decision either to commit or abort the MT. If the cluster of COs is partitioned or the main CO fails the authors use a termination protocol based on the Paxos Consensus protocol [Lamport, 1998] to elect a new main coordinator. The termination protocol succeeds only if a majority of the COs in the cluster of coordinators does not fail and also belongs to the same partition. The assumption on the mobility of the cluster of COs made here is not valid in most of ad-hoc scenarios. Targeting a more general solution, we relax this

assumption and consider a generalized arbitrary mobility model.

Similar to [Xie, 2005], we assume now that every MN in a partition knows all the members of the partition it belongs to. Later we will relax this assumption in order to keep our framework generic. Given the partition membership information, the participants in every partition elect a CO and send their votes to the elected CO which takes a pre-decision on the outcome of the MT. The pre-decision can be different from the final decision. This temporary decision is communicated to all participants within the partition. If the pre-decision is Abort, then every P-MN that receives this pre-decision can safely abort the MT. If the pre-decision is Commit, every participant in the partition should wait until all pre-decisions are collected. Alternatively, when two partitions merge, then the pre-decisions are exchanged and if all pre-decisions are collected the outcome of the MT can be safely decided since these include the votes of all participants in MT. Now all the P-MN must be informed about this outcome which can be achieved through partition-aware communication protocols similar to fragment dissemination. The correctness of the basic solution described above is assured by the partition membership assumption. If this assumption is not valid a participant can be a member of a partition but the CO of that partition is not aware about the membership of this participant and subsequently the vote of this participant can be lost, i.e., not included in any pre-decision and consequently not in the final decision.

The assumption that every MN in a partition knows all the members of its partition is crucial for the partition membership aware solution. Some works [Briesemeister and Hommel, 2002; Roman et al., 2001] addressed the problem of group membership in mobile ad-hoc environments, however a general solution for mobile environments remains a challenge. Furthermore, the blocking time of participants is often not considered and in worst case all participants may be blocked forever if one of the participants disappears forever. As shown in [Davidson et al., 1985; Skeen, 1981], there exists no non-blocking atomic commit protocol if network partitioning may occur for an unpredictable duration. Fortunately, the number of blocked participants can be minimized as we will discuss in Chapter 6. This approach based on partition membership information is independent from the mobility of nodes in contrast to [Bose et al., 2005; Böttcher et al., 2007]. However, it is based on the assumption that partition membership is available to all its members. Partitions in mobile ad-hoc scenarios are usually very dynamic as nodes may leave and join partitions arbitrarily. Therefore, acquiring the global partition membership information becomes very inefficient.

Targeting a generic approach, we propose to use a subset of the P-MNs as transaction COs to replicate the transaction coordination task on more than one MN. Furthermore, we suggest using the lifetime concept in order

to put an upper bound on the blocking time of P-MNs. These two concepts should be used without restricting the mobility of P-MNs and also without using expensive distributed systems primitives such as consensus or group membership in mobile ad-hoc environments.

4.9 Chapter Summary

In this chapter, we presented a perturbation-resilient framework to provide strict atomicity for transactions in mobile environments and investigated all the perturbations identified in Section 3.3. We presented the mechanisms and techniques used to tolerate the identified perturbations. In the next Chapters 5, 6 and 7, we progressively build a modular framework of transaction atomic commit protocols for mobile infrastructure-based, mobile ad-hoc and mobile generic environments respectively. The modular framework builds upon the mechanisms and techniques identified and presented in this chapter to construct the different atomic commit protocols.

Chapter 5

Atomic Commit for Infrastructure-based Environments

In the perturbation-resilient transaction commit framework described in Chapter 4, we *(a)* investigated and classified the perturbations in the considered mobile environment, and *(b)* presented appropriate design techniques to provide resilience to each identified class under consideration of the design requirements and issues listed in the same Chapter 4.

In this chapter, we combine the presented design techniques into a modular framework of transaction atomic commit protocols for infrastructure-based mobile environments. Since different application scenarios in mobile infrastructure-based environments show different perturbation classes, we integrate only the necessary building blocks for the most common mobile systems classes, implementing the main identified fault-tolerance and recovery techniques. For example, in a mobile system where failures are not frequent, using a protocol with sophisticated perturbation-tolerance techniques only adds unnecessary overhead to the system, thus decreasing its efficiency.

5.1 Overview of our Approach: The FT-PPTC Commit

We construct a family of atomic commit protocols for three common classes of mobile infrastructure-based environments: *(1)* a failure-free environment with the environmental constraints on nodes and links described in Section 4.4, *(2)* an environment where additional to arbitrary environmental constraints frequent network disconnections and message losses occur (network disconnections and message losses are investigated in Section 4.5 and Section 4.6 respectively), and *(3)* an environment where arbitrary node failures (explored in Section 4.7) are considered in addition to *(2)*. We emphasize here the modularity of our framework as further combinations of the building blocks to construct alternate protocols or protocol adaptations are possible. Each proposed protocol is suitable for a specific mobile system's class. Accordingly, we distinguish across the set of proposed protocols (Table 5.1):

(1) Pre-Phase Transaction Commit (PPTC), a basic protocol based on the fault-tolerant pre-phase transaction commit. This version includes a minimal set of the main design techniques a transaction commit protocol should include to cope with environmental constraints. PPTC implements mainly the concepts and techniques presented in Section 4.4.

Table 5.1: Perturbation-resilience of protocol family

	Protocol		
Building Block	PPTC	FT-PPTC	FT-PPTC-Rec
Tolerating environmental constraints (Section 4.4)	+	+	+
Decoupling (Section 4.5)	+	+	+
Tolerating message loss (Section 4.6)	-	+	+
Agent or proxy (MN-Ag) (Section 4.5)	-	+	+
Tolerating nodes failures (Section 4.7)	-	-	+

(2) Fault-Tolerant PPTC (FT-PPTC), a protocol which implements fault-tolerance in addition to resilience to environmental constraints. Therefore, FT-PPTC implements in addition to PPTC the concepts and techniques presented in Sections 4.5 and 4.6.

(3) Fault-Tolerant and Recovery PPTC (FT-PPTC-Rec), a protocol which enriches FT-PPTC with the necessary mechanisms for recovery in case of node failures, i.e., FT-PPTC-Rec implements additionally the concepts and techniques presented in Section 4.7.

Table 5.1 summarizes the building blocks of each protocol. These protocols are then compared in Section 5.6 to emphasize the impact of the various building blocks.

5.2 Base Protocol: PPTC

The PPTC protocol is our basic step towards perturbation-resilient atomic commit protocols for the mobile environment. It implements the necessary techniques to cope with the main environmental constraints described in Section 4.4. We will refer to the used techniques and covered environmental constraints while describing the protocol.

As mobile participants may need an arbitrary long time to execute their fragments, and as very few assumptions can be made regarding the performance of their wireless links, resources of fixed participants may potentially be blocked for an undefined period of time. Therefore, PPTC uses our decoupling strategy (introduced in 4.5) to *decouple* the commit of mobile participants from that of fixed participants. In the pre-commit phase (Figure 5.1), PPTC collects the votes of mobile participants to be able to reduce the commit set to a set of entities in the fixed network. The core phase involves only FNs and can be completed by any atomic commit protocol for wired networks, such as the traditional 2PC protocol. Consequently, we term this as the *core 2PC PPTC phase* as we select the established 2PC protocol to implement it. 2PC was arbitrarily chosen for this phase of PPTC especially because it is widely used in wired networks. This is not a restriction since any other established commit protocols in fixed networks can be used in this phase.

The pre-commit phase involves only P-MNs. As discussed in Section 4.4, a timeout-based concept is exploited to reach a provisional "Commit" decision at the end of the pre-commit phase (Figure 5.1). The only difference to the timeout-based concept used in this protocol is that S_t represents an estimated upper bound for the time needed to send a vote to the CO. The CO waits for the expiration of TO_{CO} and finalizes the pre-commit phase by a provisional "Commit" or an "Abort" decision. The CO proceeds to the second phase of PPTC, only if it receives "Yes" votes from all P-MNs within the specified time-limit (TO_{CO}). The transaction is aborted as soon as one P-MN aborts the transaction or TO_{CO} expires at the CO before receiving all the votes of

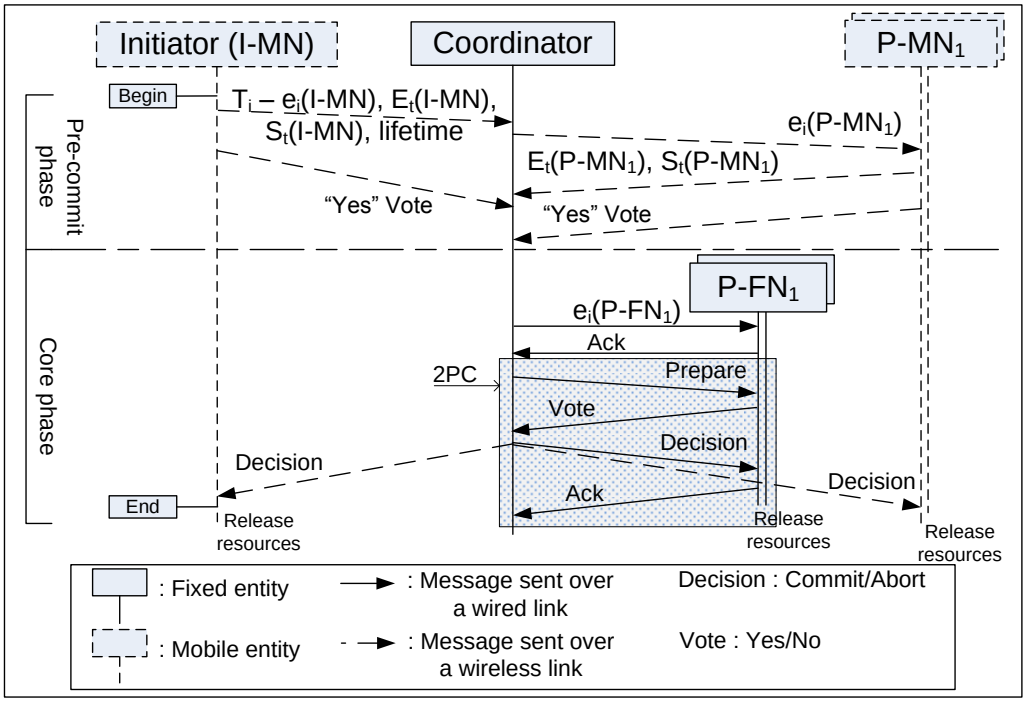

Figure 5.1: Scenario execution of the MT T_i using the PPTC protocol

P-MNs.

As a result of the pre-commit phase, the P-MNs delegate the CO to execute the 2PC protocol on their behalf. The second phase of the protocol begins, when the CO sends the execution fragments of P-FNs to their corresponding FNs and the 2PC protocol is executed to collect their votes. If all P-FNs vote for committing the MT, the CO decides to commit, otherwise it decides to abort the mobile transaction.

Figure 5.1 illustrates the execution of the PPTC protocol. The activities of each participant are outlined below. Specifically, we detail the activities of I-MN, CO and P-MN later in this chapter in Algorithm 1, Algorithm 2 and Algorithm 3 respectively.

5.2.1 Activities of the initiator mobile node

The I-MN initiates the mobile transaction T_i, extracts its execution fragment $e_i(I\text{-}MN)$, computes its E_t, S_t and $lifetime$ of the initiated transaction and sends them along with the rest of the MT $T_i - e_i(I\text{-}MN)$ to the CO (Algorithm 1). The I-MN begins the processing of $e_i(I\text{-}MN)$. Whenever the I-MN needs

Algorithm 1: I-MN's Algorithm (PPTC)

```
1  Initialize T_i;
2  Extract execution fragment e_i(I-MN);
3  Compute E_t(I-MN), S_t(I-MN) and lifetime;
4  send T_i − e_i(I-MN), E_t(I-MN), S_t(I-MN) and lifetime to CO;
5  while processing e_i(I-MN) do
6  |   if E_t(I-MN) and/or S_t(I-MN) need to be extended then
7  |   |   compute new timeout value(s);
8  |   |   send new value(s) to CO;
9  |   end
10 end
11 if I-MN decides to abort T_i then
12 |   abort T_i;
13 |   send No vote to CO;
14 |   return;
15 else                                   /* I-MN decides to commit T_i */
16 |   send Yes vote to CO;
17 |   wait for decision from CO
18 |   if decision is Commit then
19 |   |   commit T_i;
20 |   |   return;
21 |   else                                        /* decision is Abort */
22 |   |   abort T_i;
23 |   |   return;
24 |   end
25 end
```

to extend its E_t and/or S_t, it sends a message to the CO with the new timeout value(s). The I-MN sends a "No" vote to the CO whenever it decides to abort the MT. If the I-MN successfully completes the execution of its fragment, it sends a "Yes" vote to the CO. After receiving the final decision the I-MN (like the other P-MNs) is not supposed to send an "Ack" message to the CO since the PPTC protocol is not designed to tolerate message losses. P-FNs acknowledge the CO upon receiving the final decision as part of the 2PC protocol which is adopted for the core phase in PPTC.

5.2.2 Activities of the coordinator

In PPTC, the CO is the MN-Ag of the I-MN. Upon receiving $T_i - e_i(I\text{-}MN)$ from the I-MN, the CO extracts the execution fragments of the P-MNs and sends each fragment to its corresponding P-MN. The CO computes also the

Algorithm 2: CO's Algorithm (PPTC)

```
1  wait for T_i − e_i(I-MN), E_t(I-MN), S_t(I-MN) and lifetime;
2  extract and send execution fragments to the P-MNs;
3  initialize all the timeouts of the P-MNs to 0;
4  let P_m = {P-MN_1, ..., P-MN_m} the set of all the P-MNs;
5  if lifetime is undefined then
6  |  TO_COi ←
   |  max(E_t(P-MN_1) + S_t(P-MN_1), ..., E_t(P-MN_m) + S_t(P-MN_m));
7  else
8  |  TO_COi ← lifetime;
9  end
10 while waiting for TO_COi to expire do
11 |  if lifetime is undefined AND value of E_t and/or S_t (initial or
   |  extended values) of one of the MNs in P_m is received then
12 |  |  recompute TO_COi;
13 |  end
14 |  if Abort message is received from one of the MNs in P_m then
15 |  |  send Abort to all members of P_m;
16 |  |  return;
17 |  end
18 end
19 if Yes Vote is received from each P-MN then
20 |  start a 2PC protocol to collect the votes from all P-FNs;
21 |  if all votes were Yes then
22 |  |  send Commit message to all members of the commit set;
23 |  |  return;
24 |  else                          /* at least one of the votes is No  */
25 |  |  send Abort to all members of the commit set;
26 |  |  return;
27 |  end
28 else                                /* at least one Vote not received */
29 |  send Abort to all members of the commit set;
30 |  return;
31 end
```

timeout of the MT (Algorithm 2, lines 5-9). If the $lifetime$ received from the I-MN is not undefined, the timeout of the MT is set to $lifetime$. Otherwise, if the CO receives E_t and/or S_t from any P-MN, it updates its timeout (lines 11-13). The CO waits for the expiration of TO_{CO_i}. If it receives a "Yes" vote from each P-MN within this time, it distributes the execution fragments of the P-FNs along with the vote request ("Prepare" message).

If the CO receives an "Abort" message from any of the P-MNs before the expiration of TO_{CO_i} or if it doesn't receive the vote of at least one P-MN within this timeout, it sends an "Abort" message to the rest of the P-MNs and the whole transaction is aborted. After sending the execution fragments of the P-FNs, the CO starts a 2PC protocol session to collect the votes from them. If the CO receives a "Yes" vote from all the P-FNs, it decides to commit the transaction and sends "Commit" decision to all the participants. If it receives at least one "No" vote (or no reply) it decides to abort the transaction and sends "Abort" decision to all participants.

5.2.3 Activities of a participant mobile node

Upon receiving its execution fragment, the P-MN computes E_t and S_t, and sends them to the CO (Algorithm 3). The P-MN behaves then exactly like the I-MN.

Algorithm 3: P-MN's Algorithm (PPTC)

```
1 wait for receiving the corresponding execution fragment;
2 Compute Et(Par-MN) and St(Par-MN);
3 send Et(Par-MN) and St(Par-MN) to CO;
  /* continue with step 10 to step 25 of Algorithm 1 substituting
     I-MN with P-MN                                              */
```

5.2.4 Activities of a participant fixed node

P-FNs behave according to the established 2PC protocol, i.e., a P-FN executes its fragment, waits for the "Prepare" message, sends its vote and waits for the decision. Upon receiving the decision, the P-FN acknowledges the CO. Note that 2PC is not a restriction here and any further established protocol such as 3PC or Paxos Commit can be used.

The PPTC protocol copes with a wide range of environmental constraints and suits well for closed mobile systems with high coverage (using e.g. GSM or UMTS technologies), where failures can be controlled and are less probable than other systems.

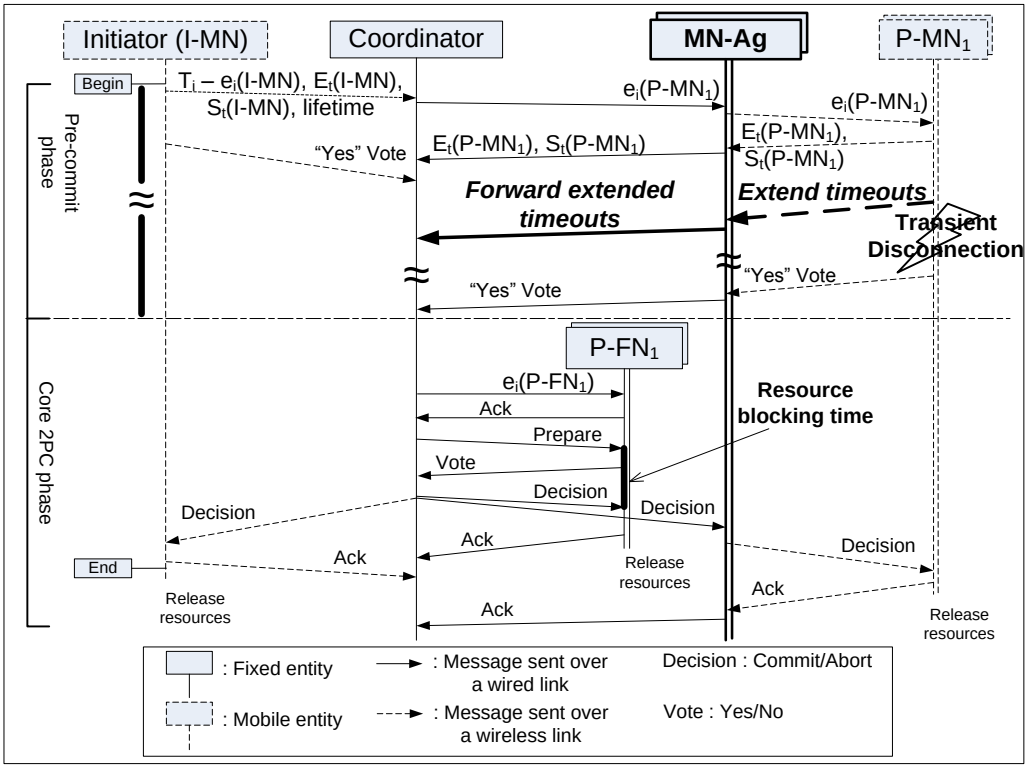

Figure 5.2: Failure-prone execution of the MT T_i using the FT-PPTC protocol (for simplicity only transient disconnections are illustrated)

5.3 Fault-Tolerant Coverage Protocol: FT-PPTC

FT-PPTC extends the PPTC protocol by adding fundamental fault-tolerance techniques presented in our framework in Sections 4.5 and 4.6 (Table 5.1). FT-PPTC tolerates in addition to environmental constraints network disconnections and message losses.

To support and improve the decoupling strategy in FT-PPTC, a MN-Ag is assigned to each P-MN as discussed in Section 4.5 (Figure 5.2). The MN-Ag in FT-PPTC is also responsible for executing the 2PC protocol on behalf of its corresponding P-MN. The MN-Ag can also take the CO role if the corresponding P-MN is the initiator of the transaction, i.e., the I-MN. As described in Section 4.5, introducing the MN-Ags simplifies the handling of the different types of communication failures (network disconnection and message loss).

5.3.1 Activities of a mobile node agent

Upon receiving the execution fragment of its corresponding P-MN from the CO, the MN-Ag forwards it to the corresponding P-MN (Algorithm 4, line 3). After receiving E_t and S_t from the P-MN, the MN-Ag forwards this information to the CO. After receiving a "Yes" or "No" vote from the P-MN, the MN-Ag forwards the vote to the CO. Upon receiving the decision from the CO, the MN-Ag forwards it to the P-MN as soon as it is available (connected to the network). After receiving the "Ack" for decision reception from the P-MN, the MN-Ag acknowledges the CO. It is key to mention that the MN-Ag is not an active participant in the execution of the MT, since it does not have to know any information about the application and does not need to process any part or fragment of the MT.

Algorithm 4: MN-Ag's Algorithm (FT-PPTC)

```
1  wait for receiving execution fragment e_i(P-MN) of the corresponding
   P-MN from CO;
2  send an estimation of the timeouts of corresponding P-MN to the CO;
3  forward e_i(P-MN) to corresponding P-MN as soon as it is available;
4  for any received message do
5      if message contains possible disconnection of the corresponding P-MN
       and its reasons then
6          recompute the timeouts based on disconnection reasons;
7          update the token with this information;
8          send extended timeouts to the CO;
9      else if message is sent by the CO then
10         forward message to the corresponding P-MN as soon as it is
           available;
11     else if message is sent by P-MN then
12         forward message to CO;
13     end
14 end
```

The MN-Ag can take some decisions on behalf of his corresponding P-MN which are discussed next. These decisions include the extension of the timeouts of the P-MN in case of a transient (predictable or unpredictable) disconnection. The MN-Ag is also given the responsibility to send an estimation of the timeouts of the corresponding P-MN direct after receiving the execution fragment of this P-MN (line 4). This estimation can be corrected after receiving new timeout values (E_t and S_t) from the P-MN.

The extension of FT-PPTC with respect to PPTC affects the I-MN and the P-MN only by requiring them to send an "Ack" message to their cor-

responding MN-Ags when they receive the final decision (Figure 5.2). This allows FT-PPTC to tolerate additional message losses compared to PPTC. Moreover, the CO communicates in FT-PPTC with the P-MNs through their corresponding MN-Ags.

Compared to PPTC, FT-PPTC is more suitable for systems using wireless technologies with lesser coverage (such as WLAN) and where failures cannot be controlled due to the openness of the system and the number of participants joining and leaving the system. FT-PPTC allows also P-MNs to disconnect during the execution of their fragments to save energy which is a major requirement in mobile environments in general. The disconnection time can also be adapted to the user needs and communicated to the corresponding MN-Ag that informs the MT CO accordingly. In order to tolerate network disconnections and message losses, FT-PPTC adds an additional overhead in terms of message complexity as compared to PPTC. This message overhead is investigated in Section 5.6.1.

5.4 Fault-Tolerant and Recovery Protocol: FT-PPTC-Rec

FT-PPTC-Rec extends FT-PPTC to tolerate node failures. The CO and MN-Ags use a combination of logging and checkpointing scheme (Section 4.7) to recover if a failure occurs during the execution of the mobile transaction.

Upon receiving $T_i - e_i(\textit{I-MN})$ from the I-MN, the CO creates a *Token* for T_i, which includes one entry for each $e_i(\textit{P-MN})$ and contains the identity of the CO and the commit set (Algorithm 5, lines 1-3). Each entry includes the state of processing of $e_i(\textit{P-MN})$ as discussed in Section 4.7 (idle, active, pre-committed, committed and aborted states). The state of $e_i(\textit{I-MN})$ is set to "active" and the state of the rest of the execution fragments is set to "idle". The entries for the execution fragments of the P-MNs additionally include their corresponding E_t and S_t. After receiving E_t and S_t from one P-MN, the CO sets the state of its corresponding fragment to "active" and updates its timeout values (E_t and S_t). If a new E_t and/or S_t (extended values) is received either from I-MN or from a P-MN, then the CO updates the *Token* also. If it receives the updates from the I-MN and a "Yes" vote from every MN-Ag involved in the MT within its computed timeout TO_{CO_i} (Algorithm 5, line 23), it stores them in the corresponding *Token* and sets the state of the MT to "pre-committed". Similar to the CO, MN-Ags implement a logging and checkpointing scheme to recover from failures (Algorithm 6).

Similar to FT-PPTC, FT-PPTC-Rec is well suited to environments hav-

Algorithm 5: CO's Algorithm (FT-PPTC-Rec)

```
1  wait for T_i − e_i(I-MN), E_t(I-MN), S_t(I-MN) and lifetime;
2  create a Token for T_i;
3  set the state of the MT to "active" in T_i's Token;
4  extract and send execution fragments of P-MNs to their corresponding
   MN-Agents;
5  initialize all the timeouts of the P-MNs with 0;
6  let P_m = {P-MN_1, ..., P-MN_m} the set of all the P-MNs;
7  if lifetime is undefined then
8  |  TO_CO_i ← max(E_t(P-MN_1) + S_t(P-MN_1), ..., E_t(P-MN_m) + S_t(P-MN_m));
9  else
10 |  TO_CO_i ← lifetime;
11 end
12 while waiting for TO_CO_i to expire do
13 |  if lifetime is undefined AND value of E_t and/or S_t (initial or extended
   |  values) of one of the MNs in P_m is received then
14 |  |  recompute TO_CO_i;
15 |  |  update the Token of T_i with the received value(s);
16 |  end
17 |  if Abort message is received from one of the MNs in P_m then
18 |  |  set the state of the MT to "aborted" in T_i's Token;
19 |  |  send Abort to all members of P_m;
20 |  |  return;
21 |  end
22 end
23 if updates are received from I-MN and a Yes vote from each MN-Ag then
24 |  write received updates to the corresponding Token;
25 |  set the state of the MT to "pre-committed" in T_i's Token;
26 |  start a 2PC protocol to collect the votes from all P-FNs;
27 |  if all votes were Yes then
28 |  |  set the state of the MT to "committed" in T_i's Token;
29 |  |  send Commit message to all members of the commit set;
30 |  |  return;
31 |  else                          /* at least one of the votes is No  */
32 |  |  set the state of the MT to "aborted" in T_i's Token;
33 |  |  send Abort to all members of the commit set;
34 |  |  return;
35 |  end
36 else     /* either the updates of the I-MN or at least one vote from
   the MN-Ag of a P-MN in P_m − I-MN are not received */
37 |  set the state of the MT to "aborted" in T_i's Token;
38 |  send Abort to all members of the commit set;
39 |  return;
40 end
```

Algorithm 6: MN-Ag's Algorithm (FT-PPTC-Rec)

```
1  wait for receiving execution fragment e_i(P-MN) of the corresponding
   P-MN from CO;
2  create a Token for e_i(P-MN);
3  set the state of e_i(P-MN) to "idle" in T_i's Token;
4  send an estimation of the timeouts of corresponding P-MN to the CO;
5  for any received message do
6      if message contains the timeouts of the P-MN then
7          update T_i's Token with the received timeouts;
8          set the state of the e_i(P-MN) to "active" in T_i's Token;
9          forward the timeouts to the CO;
10     else if message contains the updates of the corresponding P-MN then
11         update the Token with the received updates;
12         send Yes vote to CO;
13     else if message contains possible disconnection of the corresponding
       P-MN and its reasons then
14         recompute the timeouts based on disconnection reasons;
15         update the token with this information;
16         send extended timeouts to the CO;
17     else if message is sent by the CO then
18         update the Token with the received message;
19         send the received message to the corresponding P-MN as soon as
           it is available;
20     else if message is sent by P-MN then
21         update the Token with the received message;
22         send the received message to CO;
23     end
24 end
```

ing a high rate of message loss and permanent perturbations on MNs. From the message complexity point of view, FT-PPTC-Rec does not add any message overhead as compared to FT-PPTC.

5.5 Correctness Basis

According to [Bernstein et al., 1987], an atomic commit protocol has to satisfy the following five *atomicity properties or conditions*:

- *Stability:* A process cannot reverse its decision after it has reached one.

- *Consistency:* All processes that reach a decision reach the same one.

- *Validity:* The "Commit" decision can only be reached if *all* processes voted "Yes".

- *Non-triviality:* If no failure occurs and all processes voted "Yes", then the final decision should be commit.

- *Termination:* At any point in execution, if all existing failures are repaired and no new failures occur for sufficiently long time, then all processes will eventually reach a decision.

To show the correctness of the proposed family of solutions, we need to prove that each protocol satisfies the five properties listed above. It is important to notice that FT-PPTC-Rec protocol (as an extension of FT-PPTC and PPTC protocols) does not add additional functionality to FT-PPTC or PPTC but represents an improvement of some extra-functional properties of these protocols. Therefore, we only need to prove that FT-PPTC-Rec fulfills the atomicity properties. It follows directly from the specification of the FT-PPTC-Rec protocol in Section 5.4 that it satisfies the stability and the non-triviality properties. The consistency property is satisfied due to the fact that only the CO decides about the outcome of the transaction and distributes the same final decision to each participant in the commit set. Therefore, we need to prove that FT-PPTC-Rec satisfies the validity and termination properties.

Validity: We assume that the CO decides to commit the transaction when at least one of the participants has not decided yet. If this participant is a MN then the CO sends the rest of the transaction to the FNs before receiving a "Yes" vote from its corresponding MN-Ag or before receiving the updates if this MN is the I-MN. Obviously, this is in contradiction with the protocol specification. If this participant is a FN, then the 2PC protocol decides to commit the transaction before receiving all the votes from the P-FNs, which again contradicts the specification of the 2PC protocol (the same applies for any other commit protocol that can be used instead of 2PC in the wired network). In the case that at least one of the participants decides to abort the transaction, the CO cannot decide to commit the whole transaction because this decision will violate the protocol specification. Hence, the "Commit" decision can only be reached if all processes voted "*Yes*", i.e., decided to commit the transaction.

Termination: We consider any execution containing the failures listed in the perturbation model detailed in Section 3.3. For this proof we need to consider two aspects. The first aspect is whether the protocol can block at any time making all the participants waiting for an undefined amount of time for the final decision. Since our protocol is based on timeout for coordinating

the execution of the fragments of P-MNs, the CO cannot block waiting for messages from these participants. The participants also cannot block waiting for a message from the CO (which is the decision) since they are required to acknowledge this message and thus the CO is able to detect a message loss and re-send the message. The second aspect considers whether the participants are able to reach any decision after recovery or not. The P-MNs can reach a decision after recovery by asking either the CO or the corresponding MN-Ag (where the necessary state of execution of the transaction is stored) about the outcome of the transaction. For the P-FNs this is guaranteed by the recovery protocols applied there. We note that keeping the state of the execution of the MT on a stable storage allows the continuation of the execution after recovery and eventually reaching a decision.

5.6 Performance Evaluation

To evaluate the efficiency of the family of protocols developed in this work, we first qualitatively compare FT-PPTC-Rec and FT-PPTC to the existing commit protocols M-2PC [Nouali et al., 2005], CO2PC [Serrano-Alvarado, 2004; Serrano-Alvarado et al., 2004b], TCOT [Kumar et al., 2002] and UCM [Bobineau et al., 2000], regarding their perturbation resilience and message complexity. Next, we use both simulations and real experiments to investigate the performance of the PPTC protocol with respect to the throughput and resource blocking time. The perturbation resilience of FT-PPTC is also investigated using our simulation model.

5.6.1 Comparison to other Existing Approaches

We now compare the FT-PPTC-Rec protocol - which provides for maximal perturbation resilience among all other members of the protocol family introduced in Section 5.1 - to the M-2PC, CO2PC, TCOT and UCM protocols regarding the environmental constraints and failures considered in each protocol. Next, the message complexity, i.e., the number of messages exchanged during execution of these protocols, is investigated.

Perturbations Discussed in Different Protocols

In Table 5.2, we compare these protocols spanning the perturbations classified in our framework. In this table, (++) indicates a comprehensive coverage, (+) a basic coverage and (-) no coverage of the identified perturbations. FT-PPTC-Rec is the only protocol designed to handle CO failures. The com-

Table 5.2: Coverage of perturbations (++: Comprehensive, +: Basic, -: No coverage)

Protocol	**Heterogeneity**	**Transient MN Failure**	**Permanent MN Failure**	**FN and CO Failure**	**Message Loss**	**Network Disconnection**
FT-PPTC-Rec	++	++	++	++	++	++
FT-PPTC	++	+	+	+	++	++
M-2PC	+	+	-	-	++	+
CO2PC	+	+	-	-	-	+
TCOT	+	++	++	-	-	-
UCM	-	+	-	-	++	+

Table 5.3: Message complexity

Protocol	**Atomicity**	**Phases**	**Wireless message complexity**	**Overall message complexity**
PPTC	*Strict*	*2*	$(2+1)*m_p-1+e$	$\{3m_p-1+e\}+\{4f_p\}$
FT-PPTC	*Strict*	*2*	$(3+1)*m_p-1+e$	$\{4m_p-1+e\}+\{4(m_p-1)+e+4f_p+n_{ext}\}$
M-2PC	Strict	2	$(2+2)*m_p-1$	$\{4m_p-1\}+\{4f_p\}$
CO2PC	Semantic	2	$(2+1)*m_p-1$	$\{3m_p-1\}+\{4f_p+c_a\}$
TCOT	Semantic	1	$2*m_p-1+e$	$\{2m_p-1\}+\{2f_p+e_{all}+c_a\}$
UCM	Strict	1	$(1+1)*m_p$	$\{2m_p\}+\{2f_p\}$
2PC	Strict	2	$4*m_p$	$\{4m_p\}+\{4f_p\}$

prehensive handling of heterogeneity, communication and node failures distinguishes FT-PPTC-Rec and demonstrates its superiority for perturbation-resilience.

Message Complexity

We denote by e the number of timeout extensions of P-MNs and by e_{all} the number of all timeout extensions (including those of P-FNs in case of TCOT). We denote by c_a the additional costs in terms of messages for the execution of compensating transactions in case such transactions are initiated. m_p represents the number of P-MNs and f_p the number of P-FNs. We denote by n_{ext} the number of messages sent by the MN-Ag to extend the timeout of its corresponding P-MN in case of predictable and unpredictable network disconnection as discussed in Section 4.5. We adopt the message complexity of M-2PC, TCOT and UCM from the message complexity analysis published in [Bobineau et al., 2000; Nouali et al., 2005]. We refer to Figure 5.1 and Figure 5.2 to compute the message complexity of PPTC and FT-PPTC respectively. It follows that each P-MN sends two messages to the CO and receives one message from it starting from the point in time, where the votes are sent (only the I-MN sends one message and receives one). Whenever a P-MN needs to extend its timeouts (E_t and S_t) it sends an extra message to the CO. Table 5.3 does not include the message complexity of FT-PPTC-Rec since it is equal to that of FT-PPTC.

Table 5.3 details the efficiency of PPTC compared to other protocols while providing strict atomicity. We emphasize that PPTC's efficiency is comparable to classical protocols, even though it allows for fully mobile participants. Because of its importance in mobile environments we will investigate the wireless message complexity of PPTC and FT-PPTC using simulations in the next section. This investigation aims at showing the impact of the number of timeout extensions e on the overall message complexity of the evaluated protocols.

5.6.2 Simulation Methodology and Results

We present our simulation results for the common mobile system classes identified in Section 5.1. We further provide experimental results for some failure-free scenarios as implemented using a setup of multiple PDA's connected to a base laptop. Real experiments in failure-prone scenarios are planned for future work. We compare the performance of the PPTC protocol to that of M-2PC and the conventional 2PC. A comparison to CO2PC, TCOT and UCM is omitted as TCOT and CO2PC do not guarantee strict

atomicity and due to the lack of implementation details of UCM. Additionally, we highlight the perturbation-tolerance improvements provided by our various building blocks implementing the developed perturbation-tolerance techniques.

Simulation Settings

For simulation studies, we have used SimJava [SimJava], a discrete event-based simulator. We conducted our simulations using confidence intervals of 95%. For the evaluation of our protocols, we consider a wide range of parameter values to highlight the generality of our results. Table 5.4 summarizes our simulation parameters. A disconnection rate of 100% in Table 5.4 corresponds to a permanent disconnection, i.e., permanent network disconnection or permanent node failure. Disconnection rate is the ratio of time where the P-MN is disconnected from the network to the total simulation time. In some of our simulations the CO does not receive an estimation of the $lifetime$ of the MT from the initiator. In this case we say $lifetime$ is undefined (we denote it by "Lifetime=UNDEF" in the Figures). For execution times and transmission delays we used uniform distributions.

Table 5.4: Simulation parameters and settings

Parameter	**Value(s)**
#P-MNs	[1,10]
#P-FNs	[1,4]
Heterogeneity of P-MNs	Laptop, PDA, Cell phone
Fragment execution time (P-MN)	[0.3,0.4], [0.5,0.6], [0.6,0.7] s
Fragment execution time (P-FN)	[0.1,0.3] s
#Fragments per MT	n ∈ [3,15]
Heterogeneity of links	WLAN, UMTS, GSM
Transmission delay (wireless link)	[0.2,0.4], [0.4,0.7], [0.6,1.0] s
Transmission delay (wired link)	[0.01,0.03] s
Disconnection rate	0%-100%

We generate transactions as follows. We assume all transactions are of different lengths and are composed of n fragments. We distribute n uniformly across all MTs. We let some MNs initiate one transaction each at the beginning of the simulation. The number and nature of P-MNs and P-FNs are randomly selected modeling arbitrary heterogeneity. Additionally, we

implemented all the timeout strategies we developed in our framework not only for the proposed family of protocols but also for all the protocols we compared our protocols with. Our goal was to enhance the competitiveness of these protocols compared to the new ones developed in this work.

Simulation and Experimental Results in Failure-free Scenarios

For the performance analysis of transactional systems, throughput and resource blocking time are the commonly used performance metrics. We define *throughput* as the number of successfully committed MTs per time unit, and the *resource blocking time* as the time interval, where the resources at the *fixed participants* remain locked. Resource blocking time starts when the participant sends its vote to the CO and ends when the participant receives the final decision about the outcome of the MT.

Throughput: We first compare the throughput of PPTC, M-2PC, and 2PC. Figure 5.3 shows the throughput over the number of initiated transactions. We observe that throughput of PPTC is comparable to 2PC and M-2PC despite the fact that PPTC decouples the execution of fragments of P-MNs and P-FNs, whereas these fragments are executed in parallel in 2PC and M-2PC. This observation can be explained by the fact that the execution time of transaction fragments and communication delays in wired networks are considerably smaller than those in wireless networks. The time needed to execute 2PC in wireless networks can be almost neglected as compared to the time needed to collect votes from P-MNs. This applies almost for any other transaction commit protocol used instead of 2PC in the wired network.

We implemented the PPTC protocol on a testbed consisting of multiple PDAs and one laptop, which use WLAN to communicate with BSs. The throughput is comparable to results obtained from simulations. Overall PPTC displays a stable performance behavior that is similar to the behavior of the traditional 2PC protocol. This is significant given that the effect of mobile nodes is shown to be minimal for the commit operations. It also validates the effectiveness of our split two-phase approach, where the impact of the decoupling in PPTC on the performance is minimal. The matching traces of Figure 5.3 highlights that the throughput of PPTC (both in simulations and experiments) and of M2PC is competitive to 2PC. This is intuitive as our simulations here were for the failure-free scenario which is similar in some extent to a wired scenario where 2PC is actually best suited for. This also illustrates the negligible overheads of PPTC and M2PC (as compared to 2PC) for comparable failure-free cases.

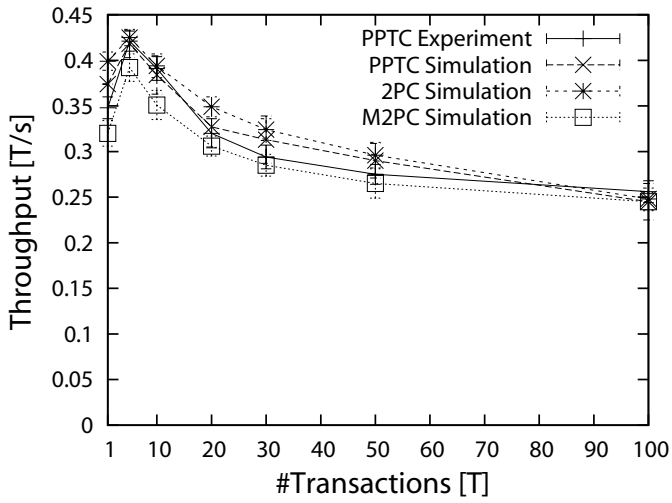

Figure 5.3: Throughput

Resource Blocking Time: Figure 5.4 depicts the blocking time over the number of initiated transactions. PPTC shows a significantly lower blocking time of the resources at P-FNs due to the decoupling of the commit of P-MNs from that of P-FNs. This decoupling makes the resource blocking time in PPTC dependent *only* on the time needed by P-FNs to execute their corresponding fragments, which is considerably shorter than the time needed by P-MNs. The results of our simulation model are also validated by the real experiments.

Heterogeneity

To evaluate the impact of heterogeneity of nodes and links on PPTC, the impact of the choice of timeouts on the commit rate is investigated. We measure the commit rate as the ratio of number of successfully committed MTs to total number of initiated MTs.

Impact of P-MNs Timeouts on Commit Rate: We conducted simulations and experiments with PPTC varying heterogeneity of links to observe how the choice of the timeout impacts the commit rate in scenarios where $lifetime$ is undefined. Figure 5.5 depicts the results of the real experiments we conducted to select an optimal timeout and those of simulations. These results show the existence of such a value for a certain given scenario.

In the case of a homogeneous network (see the curves for GSM, UMTS and WLAN networks in Figure 5.5), there is a threshold after which the

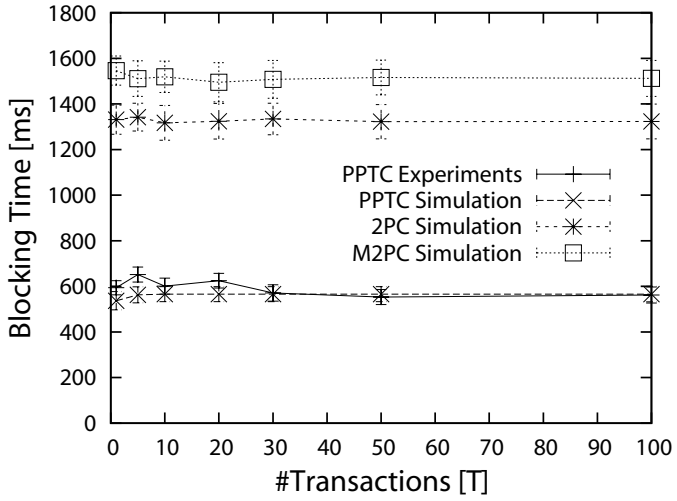

Figure 5.4: Resource blocking time at FNs

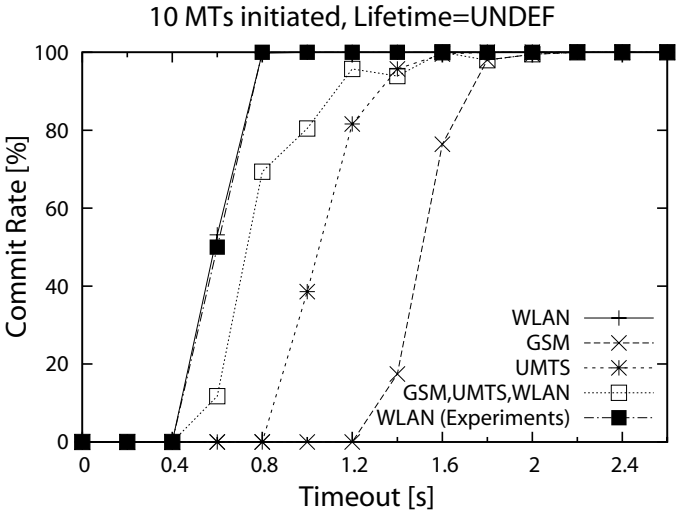

Figure 5.5: Optimal timeout selection

commit rate is 100%. This shows a deterministic behavior. In the case of a heterogeneous network (Figure 5.5) we observe an oscillation before stabilizing. This oscillation is due to the difference of delays in the different networks and also to the difference of computing capabilities of the heterogeneous P-MNs which is modeled by different fragment execution times in this simulated scenarios (Table 5.4). If the fastest P-MN initiates the transaction and the rest of the P-MNs are considerably slower, the timeout sent

to the CO by the initiator will be short in comparison to the time needed by the timeouts of the rest of the participants to reach the CO. The timeout set by the CO therefore expires before receiving any timeout from the rest of the participants and the CO aborts the transaction only due to lack of timeout estimation (Figure 4.2). Simulation results show that if an appropriate $lifetime$ is defined no oscillations result. The choice of the $lifetime$ is discussed further for the failure-prone scenario when the impact of $lifetime$ on commit rate is investigated.

Failure-Prone Scenarios

We analyze failure-prone scenarios using FT-PPTC and concentrate on four aspects: *(1)* the impact of P-MNs connectivity on commit rate, *(2)* impact of $lifetime$ on commit rate, *(3)* the impact of P-MNs connectivity on P-FNs resource blocking times, and *(4)* the impact of P-MNs connectivity on MT execution time.

Impact of P-MNs Connectivity on Commit Rate: Figure 5.6 shows that a disconnection rate of 20% leads to the abort of almost 65% of the initiated transactions if we use PPTC (or M-2PC) which is intolerable in the considered type of environments. To improve the resilience of the protocol to such failures, FT-PPTC deploys MN-Ags which are able to take some decisions on behalf of their corresponding P-MNs such as extending the timeouts if the P-MN gets disconnected. This shows a considerable improvement in the commit rate where the commit rate goes below 90% only if the disconnection rate is more than 80%. This considerable improvement comes with only a small cost of additional wireless message overhead as shown in Figure 5.7.

Impact of Lifetime on Commit Rate: Figure 5.8 illustrates the impact of increasing the $lifetime$ of the MT on the commit rate. When the $lifetime$ value is small and is in the range of the timeouts of P-MNs, FT-PPTC shows a similar behavior to PPTC. When the $lifetime$ increases the commit rate increases also since the P-MNs are now more likely to be connected within the $lifetime$ and therefore able to successfully execute their fragments and inform the MN-Ags.

Impact of P-MNs Connectivity on P-FNs Blocking Time: Figure 5.9 shows that the blocking time of P-FNs in FT-PPTC is not affected by the disconnection rate of P-MNs. However, in M-2PC the disconnection rate influences the blocking time of P-FNs since P-FNs have to wait for P-MNs as

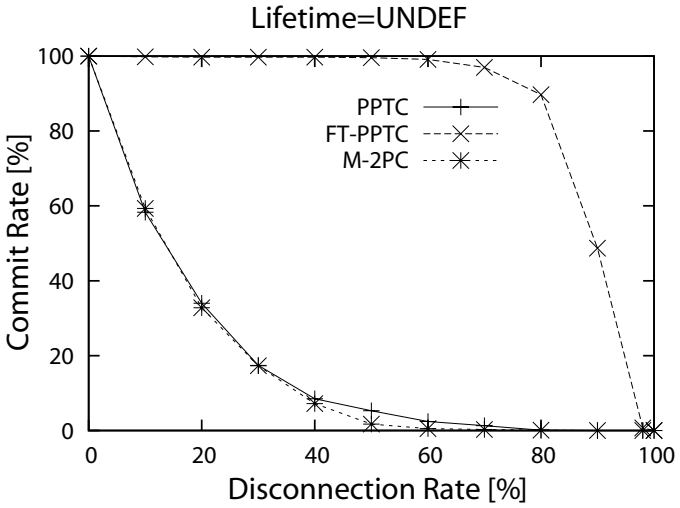

Figure 5.6: Impact of connectivity on commit rate

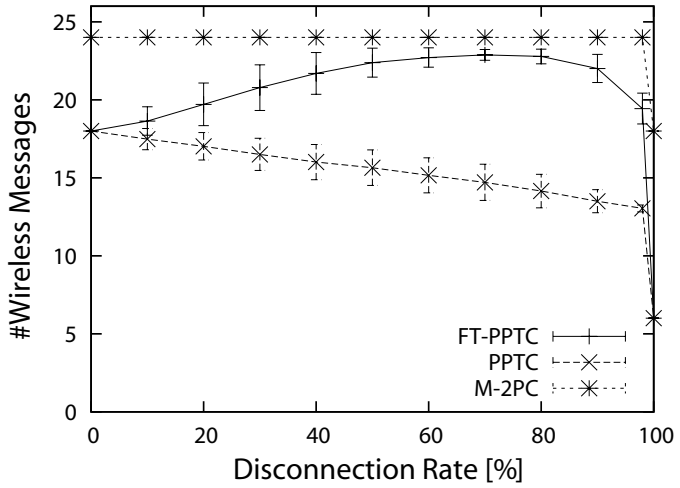

Figure 5.7: Wireless messages overhead

all participants execute their fragments in parallel. This demonstrates that FT-PPTC is highly resilient to disconnections of P-MNs as resource blocking times of P-FNs remain constant. We emphasize that resource blocking time of FT-PPTC is independent from the number of mobile participants. This highlights the scalability of our approach.

Impact of P-MNs Connectivity on MT Execution Time: We also investigated in our simulations the MT execution time, which is the time in-

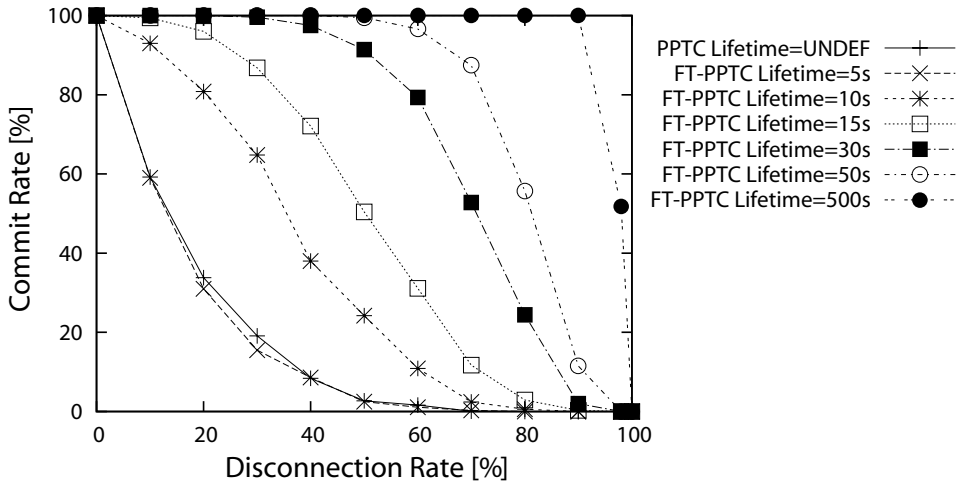

Figure 5.8: Impact of $lifetime$ on the commit rate

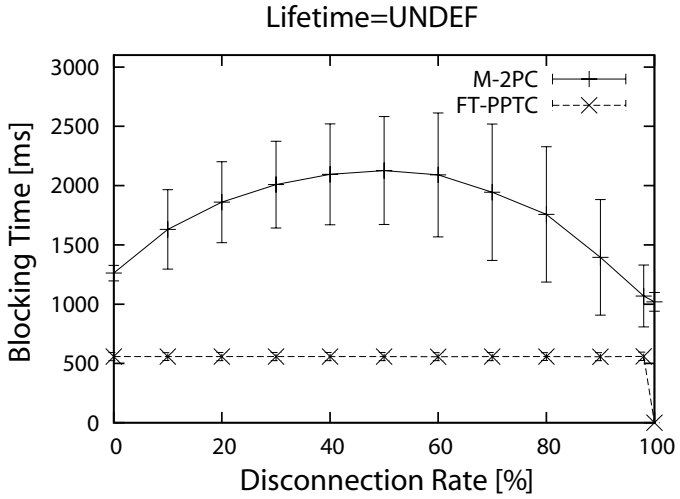

Figure 5.9: Impact of connectivity on blocking time of P-FNs

terval between the initiation of the MT and the time when the final decision is received by the last participant. Figure 5.10 shows that the decoupling does not affect the execution time considerably because the time needed to execute MT fragments on MNs and to exchange wireless messages is considerably longer than the time needed to execute MT fragments on FNs and to exchange wired messages. We conclude that the overhead in terms of MT execution time, because of decoupling is minor in comparison to the gains obtained by using this strategy in terms of perturbation resilience and re-

duction of resource blocking time of P-FNs. We observe in Figure 5.10 that the MT execution time of M-2PC is affected by the disconnection of P-MNs similarly to FT-PPTC. This is due to the fact that we implemented all the timeout strategies for both FT-PPTC and M-2PC.

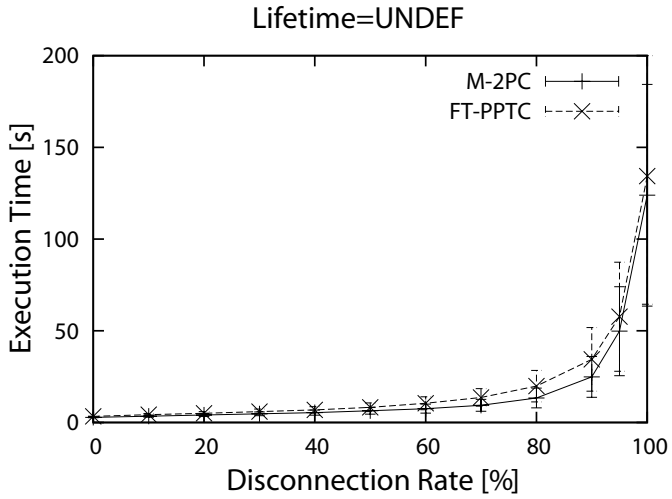

Figure 5.10: MT execution time

Discussion

We showed in our studies the utility of the different building blocks we developed in our framework. We demonstrated how these blocks can be combined during design time to adapt the atomic commit protocol to the characteristics and perturbations of the considered mobile environment. Our studies highlighted also the modularity of our approach which can be extended to satisfy the requirements of other mobile environments by identifying new buildings blocks and integrating them to the framework. The family of protocols developed in this work outperforms the existing solutions with respect to efficiency and fault-tolerance. This is achieved by adding a tolerable overhead in terms of message complexity and transaction execution times.

5.7 Chapter Summary

In this chapter, we have presented a modular framework for perturbation-resilient atomic commit protocols for mobile transactions in infrastructure-based mobile environments. We provided a family of progressive mobile

transaction commit protocols (PPTC, FT-PPTC, FT-PPTC-Rec) for three common classes of mobile infrastructure-based systems. This family of solutions is extendable and adaptable in its nature allowing the designer to customize protocol resilience to a subset of all possible perturbations based on the overall costs involved such as message complexity and charges for bandwidth utilization. We have also demonstrated, using simulations and real experiments, the efficiency, scalability and level of perturbation-resilience added by using our proposed perturbation-tolerance techniques.

In the next chapter we present our solution towards a perturbation-resilient atomic commit protocol for pure ad-hoc environments in order to extend the framework of transaction commit protocols developed throughout this thesis.

Chapter 6

Atomic Commit for Ad-Hoc Environments

In this chapter, we extend the modular framework of transaction atomic commit protocols presented in Chapter 5. So far, the atomic commit protocols framework contains only solutions for the atomic commit problem in mobile infrastructure-based environments. The extension of this framework in this chapter additionally considers perturbations that are specific to the mobile ad-hoc environment, i.e., *network partitioning*.

In mobile ad-hoc environments, network partitioning is the dominant operational case to consider. We propose a new approach that controls the decision time of MTs despite network partitioning and tolerates communication failures of P-MNs. Our approach provides for efficiency and strict atomicity in presence of network partitioning. It limits the decision time of MTs by defining a lifetime for each initiated MT. To tolerate CO unavailability and failures, a set of COs from the P-MNs of the MT is preselected allowing for the replication of the coordinator role on more than one P-MN.

6.1 Overview of our Approach: The ParTAC Commit

We propose in this chapter ParTAC, the first Partition-Tolerant Atomic Commit protocol for mobile ad-hoc environments which unlike existing protocols, *(a)* does not rely on consensus, *(b)* does not require partition membership knowledge, *(c)* is independent of the mobility patterns of mobile nodes, and *(d)* delivers best-effort transactional service availability. Table 6.1 summarizes the main advantages and contributions of ParTAC compared to the existing approaches which are detailed in Section 2.3.

Table 6.1: ParTAC vs. existing commit protocols for mobile ad-hoc environments

Protocol	Uses consensus	Uses partition/ group membership	Requires specific mobility pattern
ParTAC	*No*	*No*	*No*
[Obermeier et al., 2008]	Yes	No	No
[Bose et al., 2005] [Böttcher et al., 2007]	Yes	No	Yes
[Xie, 2005]	No	Yes	No

ParTAC adapts the lifetime concept for mobile transactions introduced in Chapter 4 to mobile ad-hoc environments in order to reduce transaction decision times. A further key idea is to use multiple coordinators and thus to replicate the coordinator role in order to tolerate both the unavailability of any subset of coordinators and communication failures. Therefore, ParTAC does not block when some of the coordinators are unavailable for a longer period of time than the transaction lifetime. Furthermore, ParTAC leverages the mobility patterns characteristic for mobile ad-hoc environments by having coordinators collect votes from other participants while moving. These votes are shared and merged by electing a single coordinator when multiple coordinators meet. Our analysis in Section 6.4 shows that ParTAC reduces the Commit/Abort decision time of initiated transactions and helps in trading-off desired levels of availability, latency, and efficiency of the transactional service by adapting protocol parameters such as the transaction lifetime and the number of coordinators.

We briefly describe the main building blocks (these building blocks are introduced in Chapter 4) of the ParTAC protocol comprising:

1. The lifetime of the MT is defined upon its initialization. The selected transaction lifetime value information along with a complete list of P-MNs and a list of preselected coordinators is communicated to every P-MN (the preselection of COs out of the P-MNs can be random or based on node properties such as IDs, mobility, connectivity, storage capabilities, etc.). However, the transaction lifetime information can only be used by the preselected COs as will be shown in the description of the protocol operations in Section 6.2. Each CO can safely abort the MT if its lifetime expires.
2. The preselected COs are required to collect votes from MT P-MNs.
3. When two COs encounter each other, they exchange their collected votes and elect a single active CO among themselves. The other CO immediately stops playing an active CO role and behaves like other normal P-MNs.
4. As a result, if all COs transitively encounter each other before the expiration of the MT lifetime, only one active CO remains which will take the final decision for the MT.

6.2 Protocol Operations

The ParTAC protocol operations of participating mobile nodes and coordinators are detailed respectively in Algorithm 7 and Algorithm 8. Since COs in ParTAC are also P-MNs, we distinguish in the following between normal P-MNs and COs. The term "P-MNs" will be used in the following to refer to P-MNs which do not play any CO role. The term "all P-MNs" includes COs as they are also P-MNs in the MT execution.

6.2.1 Activities of Participant Mobile Nodes

The activities of P-MNs are detailed in Algorithm 7. As we do not assume the existence of partition membership information, we require that P-MNs send their votes to each CO they encounter as long as there is no final decision (Algorithm 7, line 11 and lines 13-15). P-MNs know that they are encountering a CO when they receive a beacon from that CO as described in the next section (Section 6.2.2). Hence even if one CO was not aware about the P-MN's vote, e.g., due to message loss, then the vote information is not lost, but communicated to the next encountered CO. We use an acknowledgement schema in order to reduce the number of vote messages sent by one P-MN to the same CO. Therefore, if the P-MN receives an "Ack" message from a CO, it stops reacting on beacons sent by that CO (Algorithm 7, lines 16-18).

It is important to mention here that a P-MN is not allowed to change its vote once it is sent to at least one CO. So all the votes sent to the COs are the same.

Algorithm 7: P-MN's Activities in ParTAC

```
1  wait for receiving a mobile transaction T_i;
2  extract the corresponding execution fragment, the set of P-MNs and preselected
   COs;
3  let P_n = {P-MN_1, ..., P-MN_n} the set of all P-MNs;
4  let C_m = {CO_1, ..., CO_m} the set of all preselected COs;
5  start executing the received execution fragment;
6  if P-MN decides to abort T_i then
7  |  abort T_i;
8  |  send No vote to all CO in C_m;
9  |  exit;
10 else                                        /* P-MN decides to commit T_i */
11 |  send Yes vote to all CO in C_m;
12 |  while waiting for the final decision about the outcome of T_i do
13 |  |  if beacon is received from a CO then
14 |  |  |  send Yes vote to the CO from which the beacon was received;
15 |  |  end
16 |  |  if Ack is received from a CO then
17 |  |  |  stop reacting on beacons received from COs;
18 |  |  end
19 |  end
20 |  if final decision is Commit then
21 |  |  commit T_i;
22 |  |  exit;
23 |  else                                      /* decision is Abort */
24 |  |  abort T_i;
25 |  |  exit;
26 |  end
27 end
```

6.2.2 Activities of Preselected Coordinators

Algorithm 8 details the activities of preselected COs. Each CO (as it is also a P-MN in the MT) starts executing its execution fragment upon receiving the MT T_i. The preselected CO starts a timer to detect/watch the expiration of the lifetime of the MT (Algorithm 8, line 7). If it decides to vote for aborting the MT, it sends an "Abort" decision to all P-MNs of the MT. The COs periodically send presence beacons to allow other P-MNs and COs in their partition to discover their presence (Algorithm 8, line 8). These beacons

are those already being sent by the underlying routing protocol so as not to add additional messages.

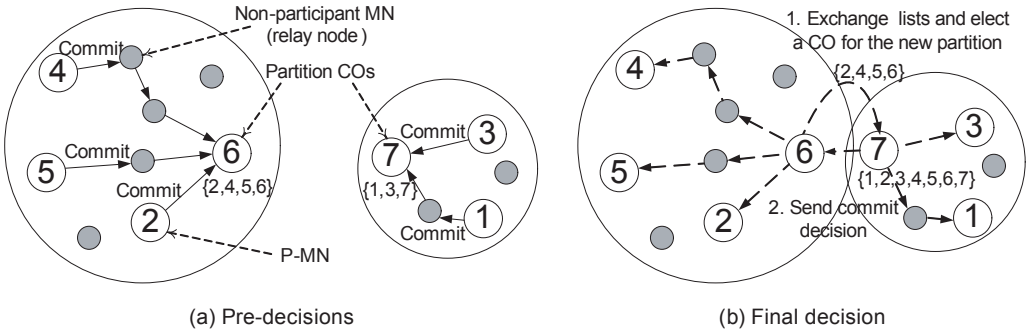

Figure 6.1: Partition-tolerant commit in mobile ad-hoc environments

Every preselected CO maintains a commit-list L of all P-MNs from which it has received a "Yes" vote. Upon receiving a "Yes" vote from a P-MN, the CO sends an "Ack" message to that P-MN to stop sending its vote to it (Algorithm 8, line 20). As an example let's consider the scenario depicted in Figure 6.1. In this scenario the COs are preselected based on their IDs (highest ID). For example, in Figure 6.1 (a), Nodes 6 and 7 are preselected as COs in the given scenario because they have the highest IDs among the P-MNs involved in the MT. Node 6 maintains in this example the commit-list $L = \{2, 4, 5, 6\}$. If the CO decides to vote for committing the MT, it adds also its ID to its own commit-list as it is also a P-MN in the MT (Algorithm 8, line 15). As soon as a CO receives a "No" vote it decides to abort the MT and sends an "Abort" decision to all P-MNs. If the lifetime of the MT expires on a CO before receiving a final decision, the CO decides also to abort the MT (Algorithm 8, lines 52-54).

If two COs encounter each other, e.g. if the corresponding network partitions join, these two COs exchange their commit-lists (Algorithm 8, lines 24-34 and 37-46) and elect one CO among themselves (in the example scenario of Figure 6.1, the CO with the highest ID, i.e., Node 7 (Figure 6.1 (b)) is elected). The other CO becomes a normal P-MN (Algorithm 8, line 35) and behaves from this point in time and onwards according to Algorithm 7. In Algorithm 8, we use a schema based on the highest ID to elect the remaining active CO, however, existing election algorithms for mobile ad-hoc environments such as [Malpani et al., 2000; Masum et al., 2006; Vasudevan et al., 2004] can also be used. COs are allowed to give their lists of votes only to other COs and only after they have completed the election process. The non-elected CO (e.g. Node 6 in Figure 6.1 (b)) sends its commit-list to the elected one (e.g. Node 7 in Figure 6.1 (b)), which merges it with its own list.

Algorithm 8: CO's Activities in ParTAC

```
1  wait for receiving a mobile transaction T_i;
2  extract the corresponding execution fragment, the lifetime of the MT, the set of P-MNs and
   preselected COs;
3  let P_n = {P-MN_1, ..., P-MN_n} the set of all P-MNs;
4  let C_m = {CO_1, ..., CO_m} the set of all preselected COs;
5  let L = ∅ the ID list of all P-MNs which sent Yes vote to the CO;
6  start executing the received execution fragment;
7  while waiting for lifetime to expire do
8      broadcast periodically beacons containing own ID;
9      if CO decides to abort T_i or receives No vote then
10         abort T_i;
11         send Abort decision to all P-MNs in P_n;
12         exit;
13     end
14     if CO decides to commit T_i then
15         add own ID to L;
16         checkList(L);
17     end
18     switch message M is received do
19         case M is a Yes vote from a P-MN
20             send Ack to P-MN;
21             add ID of sending P-MN to L;
22             checkList(L);
23         endsw
24         case M is a beacon from another CO
25             compare the received ID with the own ID;
26             if Own ID > received ID then
27                 send request to CO asking for list L (include own list L in the request);
28             else
29                 send own list L;
30                 change role to normal P-MN;
31             end
32         endsw
33         case M is a request to send list L
34             send own list L;
35             change role to normal P-MN;
36         endsw
37         case M contains a list L from another CO
38             send own list L if not already done;
39             add all IDs of received L to own list;
40             checkList(L);
41         endsw
42         case M is a Commit decision
43             commit T_i;
44             exit;
45         endsw
46         case M is an Abort decision
47             abort T_i;
48             exit;
49         endsw
50     endsw
51 end
52 abort T_i;          /* T_i is aborted if lifetime expires before reaching a decision */
53 send Abort decision to all P-MNs in P_n;
54 exit;
```

Algorithm 9: CheckList Procedure

```
1 procedure checkList(L)
2   if L contains the IDs of all P-MNs then
3     | commit T_i;
4     | send Commit decision to all P-MNs in P_n;
5     | exit;
6   end
7   return;
```

Thus, the lists are merged only if the election succeeds. If the list does not arrive at the elected CO, e.g. because of a message loss, this information is lost. The protocol can still commit the MT because the non-elected CO will send its vote to every CO it encounters after changing its role to a P-MN. The rest of the votes in the lost commit-list might have been collected by another CO since each P-MN sends its vote to every encountered CO and not to a single one of them.

The election process as described above guarantees the uniqueness of the taken decision. From the description of our approach we observe that the votes of COs can only be given to other COs after the election process. Using this schema for the election of a new and single CO guarantees that no two or more COs have the complete knowledge about which P-MNs voted to commit the MT. In the latter case these COs could take different decisions about the outcome of the MT which violates the correctness of the proposed solution.

Every time a CO election is performed, the new elected CO checks whether its new list contains all P-MNs of the MT ((Algorithm 8, line 40). If this is the case it decides to commit the MT and sends a final "Commit" decision to all P-MNs. If all P-MNs voted for committing the MT and only one CO remains for the MT, then this unique remaining CO might have a list that does not contain the IDs of all P-MNs because some votes were lost or the corresponding P-MN did not send any vote due to a transient MN failure or communication failure. In this case the expiration of the transaction lifetime will lead to a MT "Abort" decision. P-MNs share the final decisions on encounter. The final decision is inherently replicated onto the CO that turned to a P-MN since the lists of the COs are exchanged (Algorithm 8, lines 35 and 46) before electing a new CO among them. This replication is needed to recover from a failure of the last remaining CO. If the last CO fails, the rest of the P-MNs start an election algorithm to elect a new CO and if one of the P-MNs has the list of all committed P-MNs it received from the failing CO, it is elected as a new CO and the transaction is terminated.

For the dissemination of the decision and for the communication between the P-MNs inside a single partition, either flooding or a routing protocol for mobile ad-hoc environments, such as AODV [Perkins and Royer, 1999] or DSDV [Perkins and Bhagwat, 1994], are used depending on the ratio of P-MNs to non-participant MNs. The efficiency and availability of ParTAC can be enhanced by using partition-aware dissemination and routing mechanisms, such as Epidemic Routing [Vahdat and Becker, 2000] or Hypergossiping [Khelil et al., 2007].

Our proposed approach reduces the transaction decision time. Consequently the resource blocking time of P-MNs is reduced as the COs do not wait arbitrarily long to connect to decide the outcome of the MT but have bounded waiting time given by the transaction lifetime. If the transaction lifetime expires at one CO before reaching a final decision, the MT is aborted.

6.3 Correctness Basis

To show the correctness of the proposed ParTAC protocol composed of Algorithm 7 and 8, we demonstrate that it satisfies the required five *atomicity properties* [Bernstein et al., 1987]:

- *Stability:* A participant cannot reverse its decision after it has reached one.

- *Consistency:* All participants that reach a decision reach the same one.

- *Validity:* The "Commit" decision can only be reached if *all* participants voted "Yes".

- *Non-Triviality:* If no failure occurs and all participants voted "Yes", then the final decision should be "Commit".

- *Termination:* At any point in execution, if all existing failures are repaired and no new failures occur for sufficiently long time, then all participants will eventually reach a decision.

It follows directly from the specification of the ParTAC protocol in Section 6.2 that it satisfies the stability and the non-triviality properties. From the specification of the ParTAC protocol, a P-MN cannot send to two different encountered COs two different votes. Furthermore, it follows from the description of the ParTAC protocol that in the failure-free case and if all participants voted "Yes", then the final decision should be "Commit".

We now show that it also satisfies the consistency, validity and termination properties.

Consistency: The consistency property is satisfied due to the fact that only the last active CO decides about the outcome of the transaction in case the final decision is "Commit" and distributes the same final decision to every P-MN. In this case the last remaining CO is the single one which can have the final eventually complete list since at least its vote was not communicated to any other CO or P-MN according to the specification of the ParTAC protocol. If more than one CO are still remaining in the system, they can only take an "Abort" decision and no "Commit". Thus the consistency property is guaranteed by our protocol.

Validity: We assume that one of the preselected COs decides to commit the transaction when at least one of the P-MNs has not decided yet. Since this P-MN has not voted yet, its ID cannot appear in any list L (Algorithm 8, line 7) of the preselected COs according to the specification of the ParTAC protocol. Obviously, no preselected CO can then take the decision to commit the MT since this contradicts with the protocol specification (Algorithm 9, lines 1-7). In the case that at least one of the P-MNs decides to abort the transaction, the preselected COs cannot decide to commit the whole transaction because this decision will violate the protocol specification (Algorithm 8, lines 9-13). Hence, the commit decision can only be reached if all P-MNs voted "*Yes*", i.e., decided to commit the transaction.

Termination: We consider any execution containing the failures listed in the perturbation model detailed in Section 3.3. From the ParTAC protocol specification, we can observe that because we are using a timeout concept the protocol cannot block forever (the blocking of the protocol forever leads to a non-termination of the protocol). If at any point in execution all existing failures are repaired and no new failures occur for sufficiently long time, then all P-MNs will eventually reach a decision. Especially in this situation, all P-MNs (including COs) can meet each other eventually and progressively the lists of COs are filled. Therefore, the number of COs is reduced until only one CO remains having a list L containing the IDs of all P-MNs. This complete list allows this CO to take a commit decision (Algorithm 9, lines 1-7) and the protocol terminates. If the lifetime expires at any CO before reaching the final decision, the MT is aborted (Algorithm 8, lines 52-54) leading also to the termination of the protocol.

6.4 Performance Evaluation

We use simulations to validate our approach. We now present the evaluated performance metrics, the simulation model and our results that ascertain the high commit rate, the bounded decision time and the efficiency of ParTAC.

6.4.1 Methodology and Simulation Settings

For the evaluation of the ParTAC protocol, we focus on three major performance metrics: *(a) Commit rate* as it determines the service availability, *(b)* commit latency or *transaction decision time* as it determines the service response time, and *(c) message complexity* as it determines the scalability and efficiency of our approach. We measure the commit rate as the ratio of number of successfully committed MTs to total number of initiated MTs. The transaction decision time is the time needed to take a decision about the outcome of the initiated MT, i.e., the time between the initiation of the MT and the time where the final decision is reached at the CO. The blocking time of P-MNs is determined by the transaction decision time and the time needed for the final decision to reach the P-MNs. This time is dependent on the implementation of the dissemination protocol of the final decision and therefore will not be further investigated in our performance evaluation. The message complexity of ParTAC is defined as the number of messages sent and received in average by each P-MN during the execution of the MT.

The performance of our approach is evaluated based on the service delivery level assured by the protocol and defined basically by the commit rate and the decision time. The costs of the assurance of the service delivery level are measured in terms of message complexity. We focus in our performance evaluation on the impact of network partitioning on the performance metrics.

For our simulation studies we have used J-Sim [J-Sim; J-Sim Wireless], a component-based, compositional simulation environment that is entirely developed in Java and increasingly used in the mobile ad-hoc community [Tyan et al., 2009]. For the performance evaluation of the ParTAC protocol, we consider a representative range of parameter values to assess the described approach. We select the commonly used Random Waypoint mobility model [Broch et al., 1998] and the Reference Point Group Mobility (RPGM) model [Hong et al., 1999]. We fix the mobility area and the communication range, and vary the number of nodes to consider scenarios where the network is heavily partitioned and others where the number of partitions is low over time. We also vary the node speed to investigate its impact on the performance of ParTAC. We generate the mobility scenarios using the BonnMotion mobility simulator [BonnMotion]. Given its importance, for all our simula-

Table 6.2: Simulation settings

Parameter	**Value(s)**
Geographical area	$2km$ x $2km$
Communication range	$250m$
Mobility models	Random Waypoint (RWP), RPGM
Node speed	LOW uniform in [0.5, 1.5] m/s MEDIUM uniform in [3, 10] m/s HIGH uniform in [10, 25] m/s
#Nodes	$\in$ [20,400]
#COs	$\in$ {2,3,5,7,10}
#P-MNs	$\in$ {5,10,20}
lifetime	$\in$ {$5s$,$60s$,$120s$,$300s$,$900s$,1day}

tion studies we vary the partitioning degree through varying the number of nodes (note that for RPGM we need to use more nodes to reach the same partitioning degree). The partitioning degree or degree of separation is provided by BonnMotion and reflects how likely it is that two randomly chosen nodes are not within the same partition at a randomly chosen point in time.

We generate transactions of similar length and with execution fragments of P-MNs of similar length also. We initiate one transaction at the beginning of each simulation. We vary the number of P-MNs, the number of preselected COs and the lifetime to study the impact of these parameters on the performance of ParTAC. Each simulation is repeated 200 times for statistical significance of the results. Table 6.2 summarizes our simulation settings.

6.4.2 Simulation Results

We now present the results of our simulation studies for the defined performance metrics. As mentioned before, we simulate ParTAC under different network conditions and vary all protocol parameters to study the behavior of our protocol in a wide range of possible deployment scenarios. Overall, we split the results for "Abort" and "Commit" cases to have better insights to ParTAC.

Impact of Transaction Lifetime

We fix in this scenario *(a)* the number of P-MNs to 10, *(b)* the number of preselected COs to 3, *(c)* the mobility model to Random Waypoint, *(d)* the

speed to LOW and *(e)* vary the transaction lifetime value from 5 s to 1 day. We choose the number of COs to be 3 to keep the number of exchanged messages low as will be shown when the impact of the number of preselected COs will be investigated later in this section.

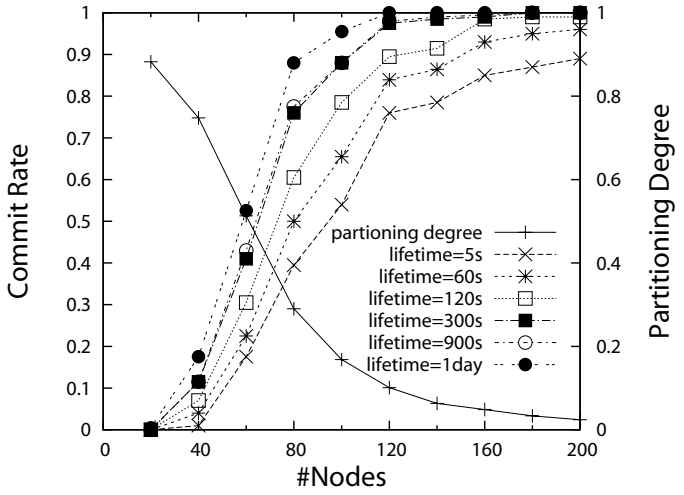

Figure 6.2: Impact of partitioning degree and lifetime on commit rate

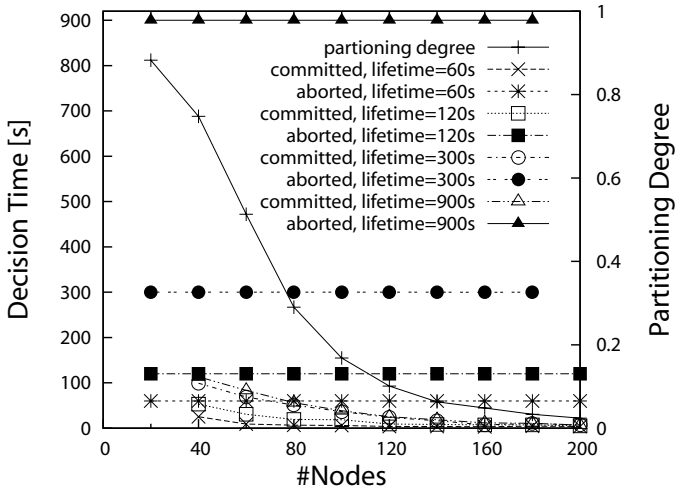

Figure 6.3: Impact of lifetime on decision time

Figure 6.2 shows how the commit rate behaves when the number of nodes or the partitioning degree varies. We observe that the commit rate is inversely

proportional to the partitioning degree. If the partitioning degree decreases the number of partitions decreases and the number of committed transactions increases. Figure 6.2 illustrates also that an increasing transaction lifetime value results in a higher commit rate. Therefore, an appropriate selection of the lifetime value is important to reach a higher commit service availability. However, reaching a higher commit service availability comes at the cost of a higher commit service latency as shown in Figure 6.3. This figure illustrates the existence of a trade-off between the commit service availability and latency. Especially in the case when the MTs are aborted, the COs need to wait for the expiration of the lifetime to abort the MT which increases the commit latency considerably.

Even in the "Abort" case the efficiency of ParTAC does not decrease as shown in Figure 6.4. The number of exchanged messages is comparable to the "Commit" case since we are using an acknowledgment schema as described in Section 6.2.1: when a P-MN receives an "Ack" message from the CO it stops sending its vote to this CO. The number of messages exchanged per P-MN during the execution of the ParTAC protocol varies between 5 and 12, which demonstrates the efficiency of our approach even in highly partitioned ad-hoc networks. We observe also in Figure 6.4 that the message complexity slightly increases if the transaction lifetime is increased because more transactions reach the "Commit" decision when we increase the transaction lifetime.

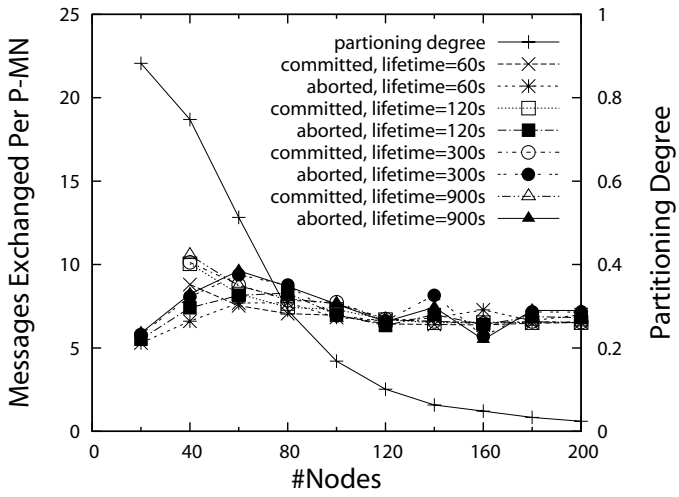

Figure 6.4: Impact of lifetime on message complexity

Our simulations show the existence of a transaction lifetime which trades off the commit rate and the transaction decision time with a moderate mes-

sage complexity. The value of the transaction lifetime is dependent on different network parameters and especially the expected partitioning level of the mobile ad-hoc environment over time.

Impact of Mobility

In this scenario, we arbitrarily fix the number of P-MNs to 10, the number of preselected COs to 3 and the transaction lifetime value to 900 s. To assess the influence of mobility on the ParTAC protocol, we vary the speed of the MNs and their mobility models. The chosen parameters are not a restriction and they are chosen to avoid combining the effect of two parameters on the studied approach in the performance evaluation scenario. This is also valid for the rest of our evaluation scenarios.

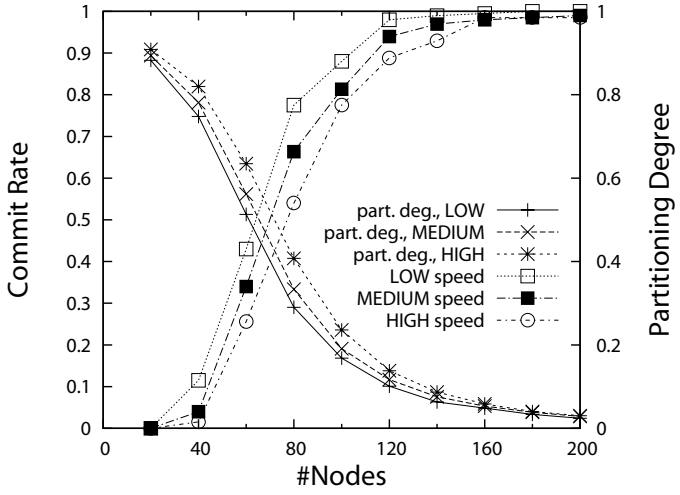

Figure 6.5: Impact of speed of MNs on commit rate

Figure 6.5 shows that the partitioning degree slightly increases if we increase the speed of MNs because of more partition split and join dynamics inside the network [Hähner et al., 2007]. This explains why the commit rate decreases if we increase the speed of MNs. Based on this observation, we conclude that the commit rate of ParTAC does not depend directly on the speed of the MNs but only on the partitioning level of the network, which can be affected by the speed of the MNs, especially, if insufficient numbers of MNs are deployed in the mobile ad-hoc scenario. Figure 6.6 illustrates that a MEDIUM speed is the best for low decision time of ParTAC because it represents the ideal case for partition splits and joins with less message loss than for the case where the speed is HIGH. Figure 6.7 shows that if the speed

of MNs increases the efficiency of the protocol decreases in the "Abort" case as compared to the "Commit" case. This can be explained by the frequent partition splits and joins if the speed increases which leads to more message loss that is compensated by the ParTAC protocol by sending more messages as described in Section 6.2.

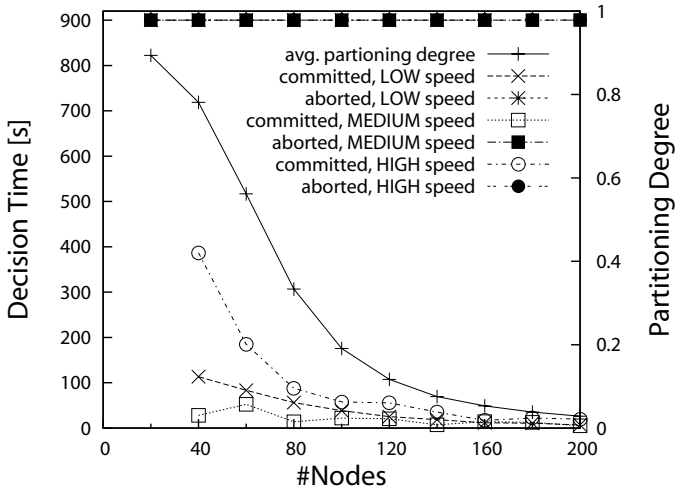

Figure 6.6: Impact of speed of MNs on decision time

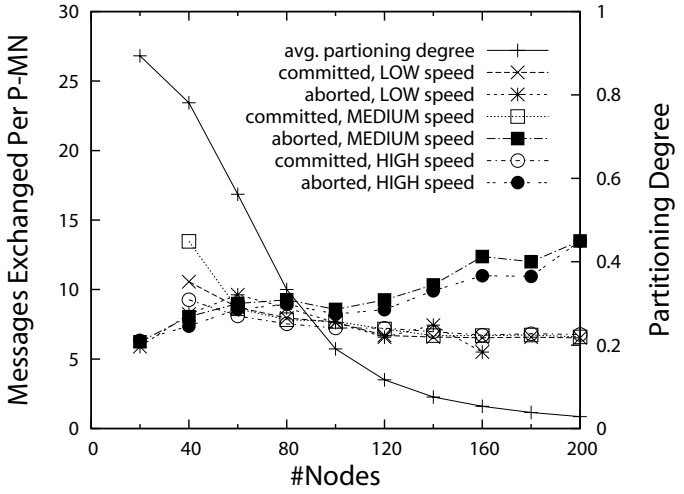

Figure 6.7: Impact of speed of MNs on message complexity

Figure 6.8 shows that the commit rate is not directly dependent on the

mobility model of the MNs. Furthermore, the number of MNs in the simulated area does not directly affect the commit rate. However, the combined effect of these two parameters is seen in the partitioning level or degree of the network and the commit rate depends on this partitioning degree as illustrated in Figure 6.8. The overhead in terms of transaction decision time (Figure 6.9) and messages exchanged between the P-MNs (Figure 6.10) is higher for RPGM than Random Waypoint. For RPGM more nodes are deployed in the same simulation area to reach similar levels of partitioning degree. This increase in the number of MNs leads to a higher message losses and higher network congestion, which explain the higher transaction decision time and higher number of exchanged messages in the case of committed transactions.

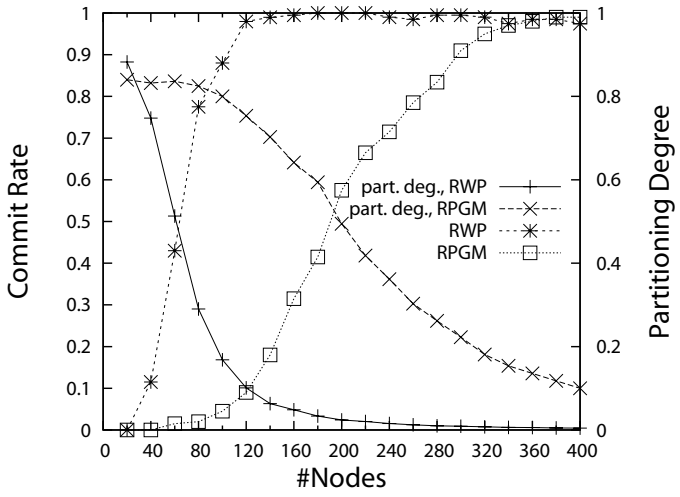

Figure 6.8: Impact of mobility model on commit rate

Based on these overall results described above, we highlight that our approach allows to efficiently reach maximal commit rates independent from the mobility pattern of the MNs (mobility model and speed). We highlight also the scalability of our approach since the increase of the numbers of nodes in the simulated area does not result in an over-proportional increase of the MT costs in term of decision time and message complexity. We emphasize here especially the efficiency of our protocol where the number of exchanged messages per P-MN remains almost constant even if the number of deployed nodes in the same area increases as illustrated in Figure 6.10.

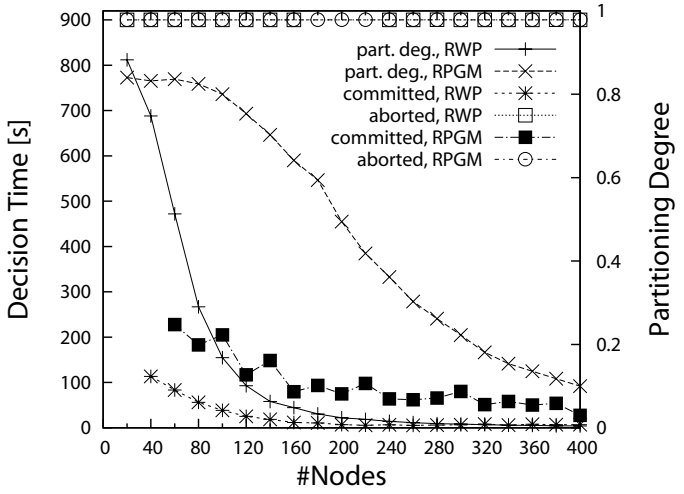

Figure 6.9: Impact of mobility model on decision time

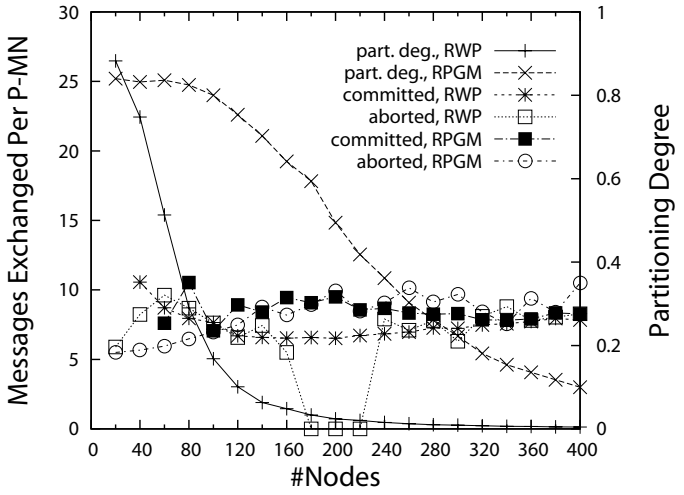

Figure 6.10: Impact of mobility model on message complexity

Impact of Number of Preselected COs

We arbitrarily fix in this scenario the number of P-MNs to 10, the transaction lifetime value to 900 s, the mobility model to RWP and the speed to LOW and vary the number of preselected COs.

The number of preselected COs does not impact the commit rate of ParTAC as illustrated in Figure 6.11. This is due to the fact that as soon as two COs encounter each other only one of them remains active and the other

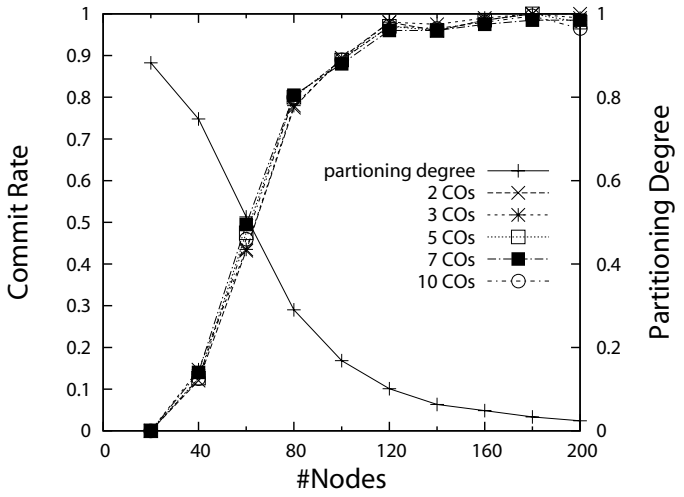

Figure 6.11: Impact of number of COs on commit rate

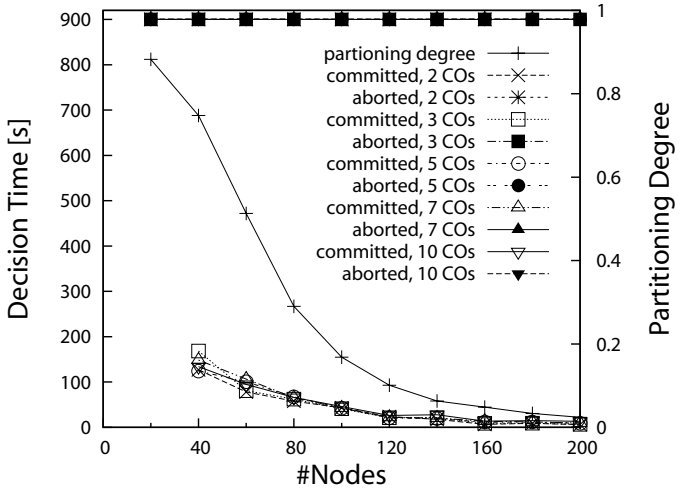

Figure 6.12: Impact of number of COs on decision time

one becomes a normal P-MN. After a certain point in time only a few (2 to 3) COs remain and all the simulated scenarios behave from this instant onwards similarly. This point in time is closer to the initiation time of the MT in the "Commit" case, as from all the COs present in one partition only one remains active as soon as they receive beacons from each other. Figure 6.12 shows that the number of preselected COs does not have an impact on the decision time also because of the same reasons given above.

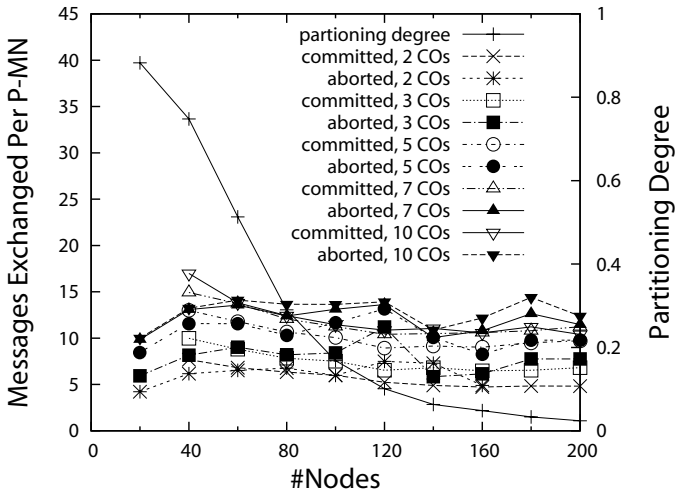

Figure 6.13: Impact of number of COs on message complexity

However, the number of preselected COs has a minor impact on the efficiency of the ParTAC protocol as shown in Figure 6.13. The slight increase of the number of messages exchanged per node is due to the fact that every P-MN needs to send its vote to more COs as the number of COs increases. It is noteworthy to mention that the main motivation for selecting higher numbers of COs is primarily to tolerate CO failures during the MT execution. Our simulations show that a higher CO failure-tolerance does only slightly impact the message efficiency.

Impact of Number of P-MNs

We arbitrarily fix in this scenario the number of preselected COs to 3, the transaction lifetime value to 900 s, the mobility model to RWP and the speed to LOW and vary the number of P-MNs.

Figure 6.14 shows, as expected, that the number of P-MNs in ParTAC influences the commit rate of the protocol. If the number of P-MNs increases, the commit rate of ParTAC decreases since more nodes need to agree on the outcome of the transaction in this case. The same applies to the decision time and message complexity as shown in Figures 6.15 and 6.16.

6.4.3 Discussion

Transactional services represent a key part of service oriented architectures and increasingly for mobile ad-hoc environments, vehicular ad-hoc networks

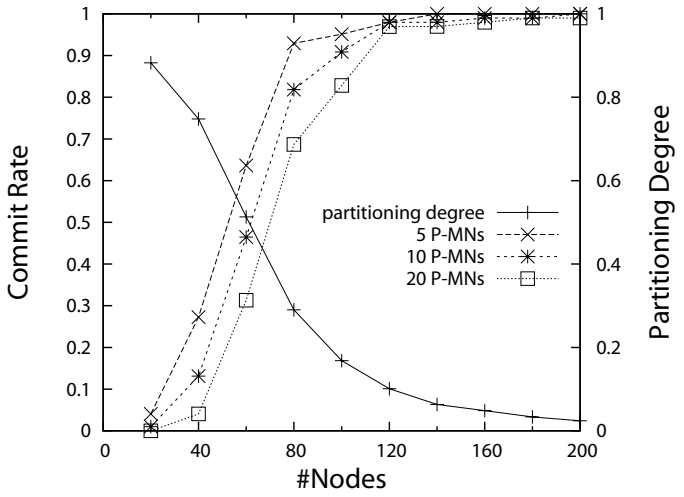

Figure 6.14: Impact of number of P-MNs on commit rate

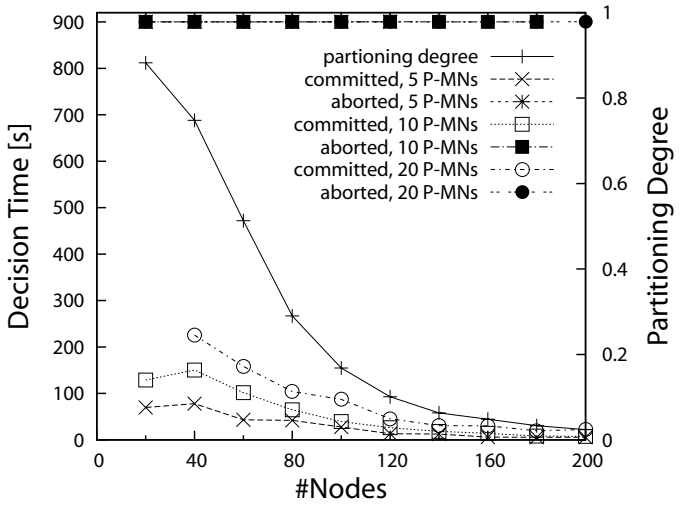

Figure 6.15: Impact of number of P-MNs on decision time

etc. Users and applications may require performing a certain number of atomic transactions with a maximized commit rate and within a certain tolerable response time. Data consistency and high transactional service availability should be provided despite the high likelihood of perturbations during the service operational conditions in mobile ad-hoc environments. Our ParTAC commit protocol considers the application requirements by defining a transaction lifetime for each initiated MT. Within the MT lifetime,

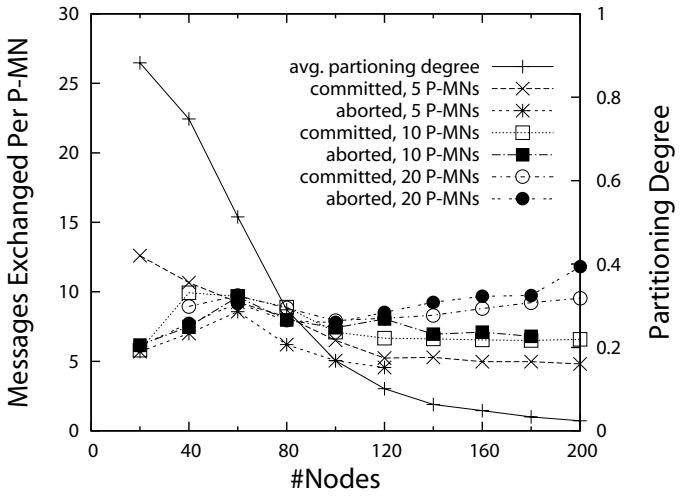

Figure 6.16: Impact of number of P-MNs on message complexity

our approach guarantees data consistency and maximizes the commit rate. This is achieved for mobile ad-hoc environments under the assumption of an arbitrary degree of perturbations with respect to network partitioning. Therefore, our commit solution provides for best effort transactional service availability for the challenging mobile ad-hoc environment. Furthermore, the ParTAC approach helps to reduce the transaction decision time resulting in a better transactional throughput and consequently in a better scalability. This allows maximizing the number of users that can for instance use the database resources on resource-limited mobile nodes.

6.5 Chapter Summary

In this chapter, we have shown how delay-awareness can help in reducing the costs of mobile transactions and in decreasing the number of aborted transactions in mobile ad-hoc environments. Delay-awareness can also help in providing perturbation-resilience in generalized mobile ad-hoc environments. We presented ParTAC, a novel atomic transaction commit protocol that provides strict atomicity in spite of frequent mobile ad-hoc environment perturbations and especially network partitioning. Our protocol is generic and especially independent from the considered mobile ad-hoc environment, as it is not based on hard assumptions such as consensus and group membership. Efficiently maximizing the commit rate while maintaining atomicity, ParTAC guarantees data consistency while allowing for high transactional

service availability and scalability.

In the next chapter, we combine our solutions for infrastructure-based mobile environments from Chapter 5 with the solution presented in this chapter to further extend our framework of transaction commit protocols developed throughout this thesis. Our objective in the next chapter is to provide an integrated commit solution for the mobile generic environment where some of the mobile devices might have access to the infrastructure.

Chapter 7

Atomic Commit for Generic Environments

In this chapter, we extend the modular framework of transaction atomic commit protocols presented in Chapters 5 and 6. Until now, the atomic commit protocol framework provides only solutions for the atomic commit problem in mobile infrastructure-based and mobile ad-hoc environments. The extension of the framework in the chapter at hand considers the mobile generic environment defined in Chapter 3. Therefore, we investigate (in this chapter) the suitability of existing commit approaches to the mobile generic environment and how we can integrate them in order to provide an atomic commit solution for this environment which:

(i) takes advantage of accessing infrastructures, by choosing reliable infrastructure nodes for coordination of transactions and by using this infrastructure to replicate commit data of mobile participants in order to tolerate network perturbations, and

(ii) tolerates network partitioning and delivers best-effort results – in terms of commit rate, message efficiency and "Commit"/"Abort" decision time – if the access to wired infrastructure is unavailable.

We introduce in this chapter the Perturbation-Resilient Transaction Atomic Commit (PeRTAC) approach, our solution for generic mobile environments. PeRTAC is an efficient integration of the FT-PPTC and ParTAC approaches described in Chapters 5 and 6 respectively. PeRTAC can be illustrated as a generalization of the FT-PPTC approach in the case that P-MNs are additionally able to communicate with each other in ad-hoc mode (beyond their ability to communicate with BSs) and a generalization of ParTAC on the involvement of P-FNs and in case P-MNs are additionally able to communicate with BSs (beyond their ability to communicate in ad-hoc mode).

7.1 Overview of our Approach: The PeRTAC Commit

Our primary goal is to offer efficient perturbation-resilience to atomic commit protocols in generic mobile environments. We start with analyzing existing approaches to identify their limitation in providing a perturbation-resilient commit solution in generic mobile environments. As proven in Chapter 5, FT-PPTC is the most suitable transaction atomic protocol for infrastructure-based mobile environments which *(a)* satisfies our efficiency requirements with respect to reduction of blocking time of FN resources (presented in Chapter 4), *(b)* copes with a wide range of perturbations encountered in mobile environments, *(c)* is delay-aware, and *(d)* implements traditional atomic commit protocols such as 2PC/3PC in pure wired environments. As shown in Chapter 6, ParTAC is a generic atomic commit solution for mobile ad-hoc environments which is designed to tolerate network partitioning in an efficient manner compared to other existing solutions. Accordingly, it is natural to integrate these two approaches to build a perturbation-resilient, delay-aware and efficient solution for generic mobile environments. Figure 7.1 (with the same legend as Figure 3.2 in Chapter 3) summarizes our objective and shows that the integration of these two protocols is not trivial. From this figure, we can observe that our main objective is to provide a solution which transforms to FT-PPTC if deployed in mobile infrastructure-based environments and to ParTAC if deployed in mobile ad-hoc environments. The shaded area in Figure 7.1 c) which is not covered by any protocol represents the gap/challenge for integrating FT-PPTC and ParTAC. The nodes in this shaded area are able to communicate with BSs and ad-hoc with other MNs, so the major task in this chapter is to define how these nodes should behave in order to fulfill our atomic commit requirements and how the behavior of these nodes influences the other participating nodes.

We start by briefly describing the strength of FT-PPTC and ParTAC in infrastructure-based and infrastructure-less environments respectively, along with their limitations in generalized network settings. Next, we describe the PeRTAC commit approach.

As illustrated in Figure 7.2, FT-PPTC decouples the commit of P-MNs from that of P-FNs. The execution of the transaction is therefore split in *two phases*. In the first phase, called the *pre-commit phase*, "sufficient" information from mobile participants is collected in order to reduce the commit set to a set of fixed nodes. In the second phase the commit involves only FNs and thus can be simply completed by any atomic commit protocol from wired networks, such as the established 2PC protocol. We refer to the sec-

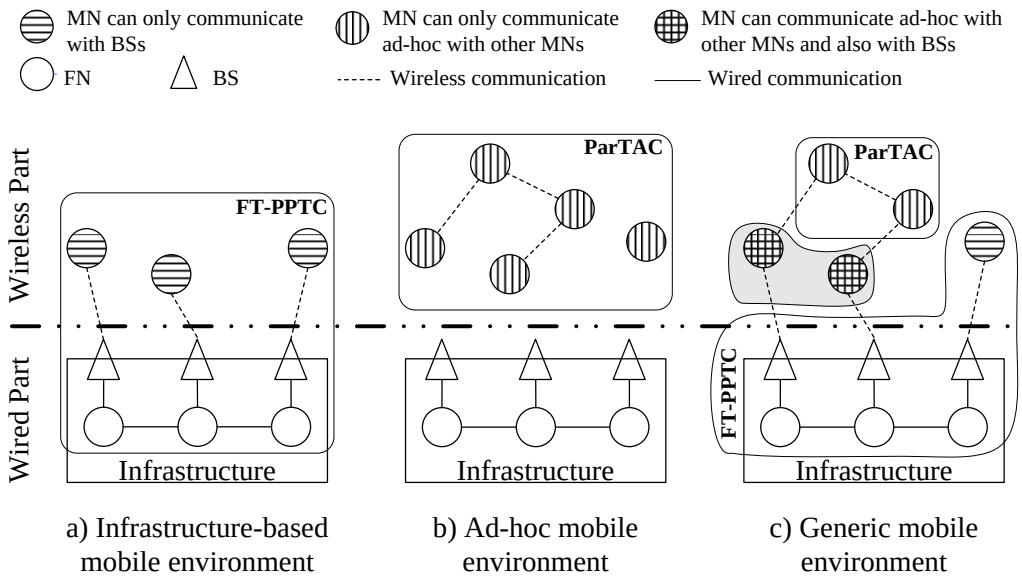

Figure 7.1: Objectives of the proposed approach

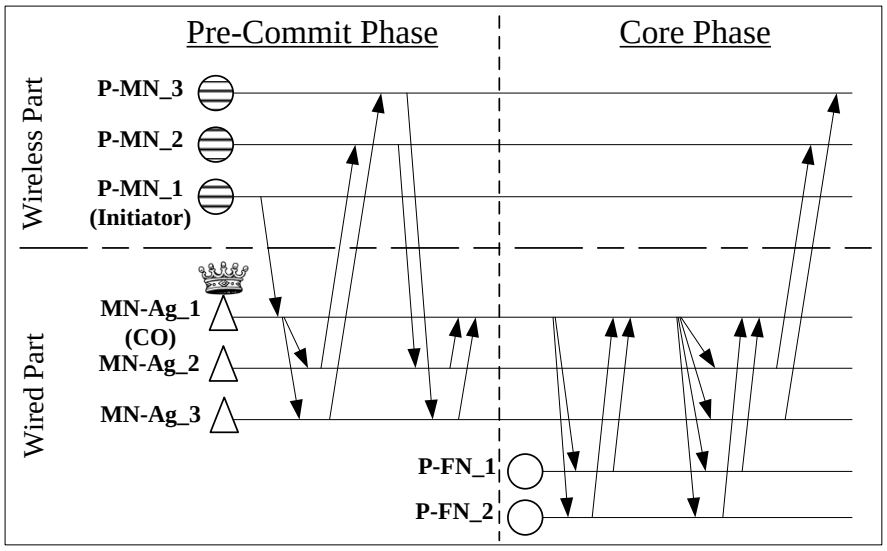

Figure 7.2: FT-PPTC flow diagram

ond phase as the *core phase*. To allow for this decoupling, a mobile node agent is assigned to each P-MN. An MN-Ag is a logical entity representing the P-MN in the wired network as described in Section 4.5. The MN-Ag is responsible for storing all the information related to the state of MTs involving the MN and is also responsible for executing the 2PC protocol on behalf of its corresponding P-MN. As shown in Chapter 5, decoupling reduces the

blocking time of the resources at the FNs. It also simplifies the handling of the different kinds of failures that rise from the mobility of MNs.

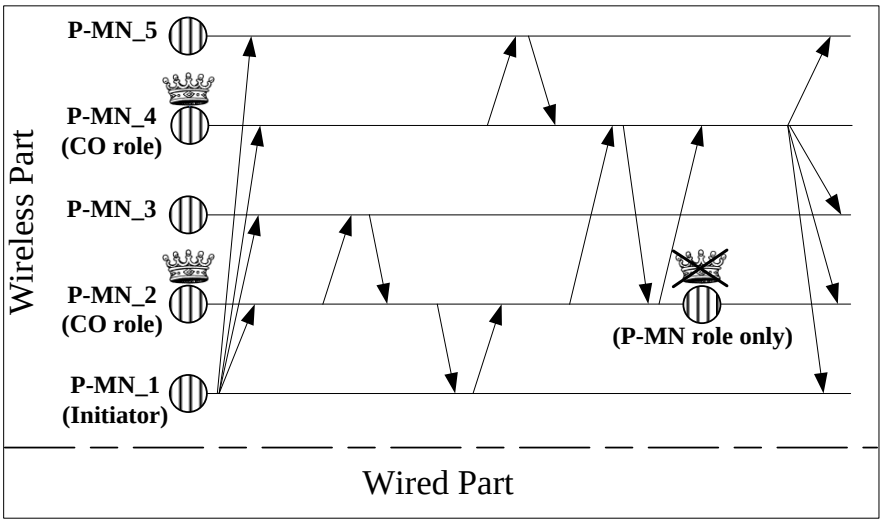

Figure 7.3: ParTAC flow diagram

Figure 7.3 shows the main building blocks of the ParTAC protocol. As described in Chapter 6 a set of coordinators in ParTAC is pre-selected among the P-MNs (the pre-selection of COs out of the P-MNs can be random or based on node properties such as IDs, mobility, connectivity, storage capabilities). Every preselected CO can safely abort the MT upon expiration of the MT lifetime if no decision is reached by then. The preselected COs collect votes from MT P-MNs. When two COs encounter each other, they exchange their collected votes and elect a single active CO among themselves. The other CO immediately stops playing an active CO role and behaves like other normal P-MNs. As a result, if all COs transitively encounter each other before the expiration of the MT lifetime, only one active CO remains which will take the final decision for the MT.

Table 7.1 summarizes why FT-PPTC and ParTAC fail in delivering a solution for generic environments and briefly highlights our requirements on the new PeRTAC approach. Transaction commit protocols developed for infrastructure-based mobile environments in general and FT-PPTC, as a representative, consider only the choice of mobile transaction COs as FNs because of the availability and perturbation-resilience of these nodes compared to MNs. These works [Bobineau et al., 2000; Kumar et al., 2002; Nouali et al., 2005; Serrano-Alvarado, 2004] do not consider at all the choice of MNs as transaction COs because of the nature of the mobile environment. To apply these approaches in mobile ad-hoc environments as a special case

Table 7.1: Requirements on the new PeRTAC approach

Protocol	**Infrastructure-based mobile environment**	**Ad-hoc mobile environment**	**Generic mobile environment**
FT-PPTC (Chapter 5)	Perturbation-resilient, efficient solution	Fails due to lack of infrastructure	Works only if the MT initiator can connect to an infrastructure
ParTAC (Chapter 6)	Not efficient due to extra overhead and long blocking time of FNs	Perturbation-resilient, efficient solution	Not efficient especially in case the access to infrastructure is available
PeRTAC	Perturbation-resilient, efficient solution	Perturbation-resilient, efficient solution	Perturbation-resilient, efficient solution

of generic environments, the choice of the CO becomes a challenge as described in Chapter 4. Not only MNs are not having stable storages to rely on them for the complete time of transaction execution but also their availability and reachability is not guaranteed even for small periods of time as opposed to FNs. Subsequently, these developed solutions simply fail in ad-hoc environments mainly because of the lack of infrastructure. In generic mobile environments these approaches might work only in a limited number of cases where the initiator of the MT can communicate with BSs and is able to connect to the infrastructure to select a FN to play the CO role. Unfortunately, this solution is in many scenarios inefficient since all the traffic between the CO and P-MNs will flow in this case through the mobile initiator.

If we consider now transaction commit protocols developed for ad-hoc mobile environments in general and ParTAC as a novel representative, we observe that COs can only be chosen among MNs as no FN is available. These approaches rely mainly on choosing more than one MN to play the CO role. In these works, the vulnerabilities of the MNs are masked by replicating the CO role. Though these commit protocols can be deployed in infrastructure-based environments as a special case of generic ones, they are inefficient due to an unnecessary additional overhead caused by the replication of the CO role and also long blocking time of FNs participating in MT execution (these approaches does not consider blocking times of FNs since they are designed for pure ad-hoc environments). In generic mobile environments these protocols are not efficient especially when some transaction participants are FNs or in case all participants are MNs but the access to the infrastructure

is available.

Targeting an atomic transaction solution which covers all three types of environments as illustrated in Table 7.1, we propose the PeRTAC protocol as a solution for generic mobile environments. This solution delivers the same performance and perturbation-tolerance of ParTAC and FT-PPTC in ad-hoc and infrastructure-based mobile environments respectively. Figure 7.4 shows that the PeRTAC protocol inherits from the FT-PPTC (Figure 7.2) its two phases, i.e., the pre-commit phase and the core phase. In the pre-commit phase, PeRTAC follows the multiple CO strategy of ParTAC (Figure 7.3) with one main difference: the multiple CO strategy in PeRTAC favors P-MNs which can connect to the infrastructure while electing a new CO on encounter as described in Chapter 6. If two COs, that are only able to communicate in ad-hoc mode, encounter each other then the election is done as described in ParTAC in Chapter 6, i.e., the one having the highest ID is elected. But if one CO that is able to access the infrastructure encounters a CO that is only able to communicate in ad-hoc mode, the former is elected and its MN-Ag becomes the CO of the MT in the wired part of the environment. In case two MN-Ags are selected in the above described way to be COs, then any election algorithm for wired networks, such as [Chang and Roberts, 1979; Garcï¿½a-Molina, 1982], might be used.

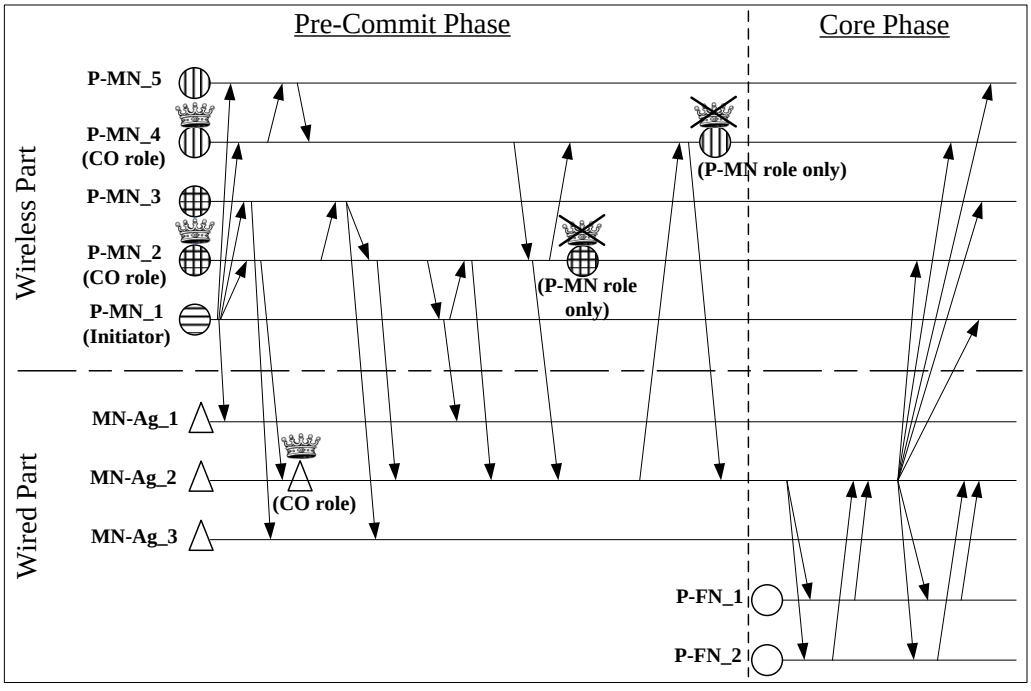

Figure 7.4: PeRTAC flow diagram

7.2 Protocol Operations

We detail in the following the protocol operations of PeRTAC that describe the activities of P-MNs, COs and MN-Ags respectively in Algorithm 10, Algorithm 11 and Algorithm 13. P-FNs activities are described in Subsection 7.2.4.

7.2.1 Activities of Participant Mobile Nodes

The activities of P-MNs are detailed in Algorithm 10. Upon receiving its execution fragment, the P-MN computes E_t and S_t (defined in Chapter 4) if it can communicate with a BS, and sends them to the corresponding MN-Ag (Algorithm 10). E_t and S_t are estimations of execution and shipping times respectively. Execution time is the time needed to execute the fragment on the P-MN and shipping time is the time needed to prepare and send the vote message to the MN-Ag. The P-MN begins the processing of its execution fragment $e_i(P\text{-}MN)$. Whenever the P-MN needs to extend its E_t or S_t and it has access to the infrastructure, it sends a message to its MN-Ag with the new timeout value(s) (Algorithm 10, lines 5-8). If the P-MN can communicate with a BS, it sends a "No" vote to its MN-Ag whenever it decides to abort the MT and its updates if it successfully completes the execution of its fragment. In case the P-MN is not able to access the infrastructure, it sends its vote to each CO it encounters as long as no ACK is received from that CO and there is no final decision (Algorithm 10, line 29 and lines 31-36). P-MNs know that they are encountering a CO when they receive a beacon from that CO as described in Section 7.2.2. Hence even if one CO was not aware about the P-MN's vote, e.g., due to message loss, then the vote information is not lost, but communicated to the next encountered CO. It is noteworthy that a P-MN is not allowed to change its vote once it has been sent to a CO.

7.2.2 Activities of Coordinators

Algorithm 11 details the activities of PeRTAC COs. Upon receiving a MT T_i, a CO creates a *Token* for the received MT, which includes all information about the participants and COs of the MT and the state of execution. The possible execution states are: Idle, active, pre-committed, committed or aborted. The state of T_i is set to "active". If a CO is a P-MN in the MT, it starts executing its execution fragment upon receiving the MT T_i. Every CO starts a timer to detect/watch the expiration of the lifetime of the MT (Algorithm 11, line 9). If a CO receives initial or updated E_t or S_t from one P-MN, the lifetime of the MT is updated if mandatory, i.e., if the lifetime

Algorithm 10: P-MN's Activities in PeRTAC

```
1  wait for receiving a mobile transaction T_i;
2  extract the corresponding execution fragment e_i(P-MN), the set of P-MNs and
   preselected COs;
3  let P_n = {P-MN_1, ..., P-MN_n} the set of all P-MNs;
4  let C_m = {CO_1, ..., CO_m} the set of all preselected COs;
5  if P-MN has MN-Ag then
6  |   Compute E_t(P-MN), S_t(P-MN);
7  |   send E_t(P-MN), S_t(P-MN) to the corresponding MN-Ag;
8  end
9  start executing the received execution fragment;
10 while processing e_i(P-MN) do
11 |   if (P-MN has MN-Ag) and (E_t(P-MN) or S_t(P-MN) need to be extended)
   |   then
12 |   |   compute new timeout value(s);
13 |   |   send new value(s) to the corresponding MN-Ag;
14 |   end
15 end
16 if P-MN decides to abort T_i then
17 |   abort T_i;
18 |   if P-MN has MN-Ag then
19 |   |   send No vote to corresponding MN-Ag
20 |   else
21 |   |   send No vote to all COs in C_m;
22 |   |   exit;
23 |   end
24 else                                          /* P-MN decides to commit T_i */
25 |   write updates to the local log;
26 |   if P-MN has MN-Ag then
27 |   |   send updates to the corresponding MN-Ag;
28 |   else
29 |   |   send Yes vote to all COs in C_m;
30 |   |   while waiting for the final decision about the outcome of T_i do
31 |   |   |   if beacon is received from a CO then
32 |   |   |   |   send Yes vote to the CO from which the beacon was received;
33 |   |   |   end
34 |   |   |   if Ack is received from a CO then
35 |   |   |   |   stop reacting on beacons received from that CO;
36 |   |   |   end
37 |   |   end
38 |   end
39 |   if final decision is Commit then
40 |   |   commit T_i;
41 |   |   exit;
42 |   else                                      /* decision is Abort */
43 |   |   abort T_i;
44 |   |   exit;
45 |   end
46 end
```

Algorithm 11: CO's Activities in PeRTAC

```
1  wait for receiving a mobile transaction T_i;
2  extract the corresponding execution fragment if the CO is a P-MN or the fragment of its
   corresponding P-MN if it is a MN-Ag, the lifetime of the MT, the set of P-MNs and COs;
3  create a Token for T_i ;
4  set the state of the MT to "active" in T_i's Token;
5  let P_n = {P-MN_1, ..., P-MN_n} the set of all P-MNs;
6  let C_m = {CO_1, ..., CO_m} the set of all preselected COs;
7  let L = ∅ the ID list of all P-MNs which sent Yes vote to the CO;
8  start executing the received execution fragment if CO is a P-MN;
9  while waiting for lifetime to expire do
10     if CO is a P-MN then
11         broadcast periodically own ID;
12     else                                                    /* The CO is a MN-Ag */
13         if value of E_t or S_t (initial or extended values) of one of the P-MNs is received then
14             update lifetime value only if it needs to be increased;
15             update the Token of T_i with the received value(s);
16         end
17     end
18     if CO decides to abort T_i or receives No vote then
19         abort T_i;
20         send Abort decision to all P-MNs in P_n; exit;
21     end
22     if CO decides to commit T_i or receives updates from corresponding P-MN then
23         add own ID to L if CO is a P-MN or add ID of corresponding P-MN if CO is a
           MN-Ag;
24         checkList(L);
25     end
26     switch message M is received do
27         case M is a Yes vote from a P-MN
28             send Ack to P-MN;
29             add ID of sending P-MN to L;
30             checkList(L);
31         endsw
32         case M is a beacon from another CO
33             compare the received ID with the own ID;
34             if (both COs are either P-MNs or MN-Ags and Own ID > received ID) or (CO
               is MN-Ag and other CO – from which the beacon is received – is a P-MN) then
35                 send request to CO asking for list L (include own list L in the request);
36             else
37                 send own list L;
38                 change role to normal P-MN or MN-Ag;
39             end
40         endsw
41         case M is a request to send list L
42             send own list L;
43             change role to normal P-MN;
44         endsw
45         case M contains a list L from another CO
46             send own list L if not already done;
47             add all IDs of received L to own list;
48             checkList(L);
49         endsw
50         case M is a Commit decision
51             commit T_i; exit;
52         endsw
53         case M is an Abort decision
54             abort T_i; exit;
55         endsw
56     endsw
57 end
58 abort T_i;          /* T_i is aborted if lifetime expires before reaching a decision */
59 send Abort decision to all P-MNs in P_n;
```

Algorithm 12: CheckList Procedure

```
1  procedure checkList(L)
2    if (L contains the IDs of all P-MNs) and (commit set contains P-FNs) then
3      set the state of the MT to "pre-committed" in T_i's Token;
                                                  /* Starting of the Core Phase */
4      start a 2PC protocol to collect the votes from all P-FNs;
5      if all votes were Yes then
6        commit T_i;
7        set the state of the MT to "committed" in T_i's Token;
8        send Commit message to all members of the commit set;
9        return;
10     else                                  /* at least one of the votes is No  */
11       abort T_i;
12       set the state of the MT to "aborted" in T_i's Token;
13       send Abort to all members of the commit set;
14       return;
15     end
16   else if (L contains the IDs of all P-MNs) then
17     commit T_i;
18     send Commit decision to all P-MNs in P_n;
19     exit;
20   end
21   return;
```

needs to be increased as described in Chapter 5. The updated information is stored in the *Token* of the CO which received that information. If a CO decides to vote for aborting the MT and it is a P-MN or receives a "No" vote, it sends an "Abort" decision to all P-MNs of the MT. The COs periodically send presence beacons to allow other P-MNs and COs in their partition to discover their presence (Algorithm 11, line 11). These beacons are those already being sent by the underlying ad-hoc routing protocol to avoid additional wireless messages. Every preselected CO maintains a commit-list L of all P-MNs from which it has received a "Yes" vote. If the CO is a P-MN and decides to vote for committing the MT or if it is a MN-Ag and receives the updates of its corresponding P-MN, it adds also its ID to its own commit-list (Algorithm 11, line 23). As soon as a CO receives a "No" vote it decides to abort the MT and sends an "Abort" decision to all P-MNs. If the lifetime of the MT expires on a CO before receiving a final decision, the CO decides also to abort the MT (Algorithm 11, lines 58-59).

If two COs encounter each other (e.g., if the corresponding network partitions join) these two COs exchange their commit-lists (Algorithm 11, lines 32-42 and 45-46) and elect one CO among themselves, e.g., based on highest ID (Algorithm 11). The other CO becomes either a normal P-MN or a MN-Ag

(Algorithm 11, line 43) and behaves from this point in time and onwards according to Algorithm 10 or Algorithm 13 respectively. If one of the COs is a MN-Ag, it is elected automatically and if both are MN-Ags the highest ID schema or another election algorithm can be used. COs are allowed to give their list of votes only to other COs and only after they complete the election process. The non-elected CO sends its commit-list to the elected one that merges it with its own list. Thus, the lists are merged only if the election succeeds. If the list does not arrive at the elected CO, e.g., due to message loss, PeRTAC can still commit the MT since the non-elected CO will send its commit-list L to every CO it encounters after changing its role to a P-MN or it will send the commit-list to the elected FN CO if it is a MN-Ag.

The election process as described above guarantees the uniqueness of the taken decision. From the description of our approach we observe that the votes of COs can only be given to other COs after the election process. Using this schema for the election of a new and single CO guarantees that no two or more COs have the complete knowledge about which P-MNs voted to commit the MT. In the latter case these COs could take different decisions about the outcome of the MT which violates the correctness of the proposed solution.

Each time a CO election is performed, the new elected CO checks whether its list contains all P-MNs of the MT (Algorithm 12). If this is the case, it sets the state of the MT to "pre-committed" and starts a 2PC session to collect the votes from P-FNs if any. If the CO receives a "Yes" vote from all the P-FNs, it decides to commit the transaction and sends "Commit" decision to all the participants. If it receives at least one "No" vote (or no reply) it aborts the transaction and sends "Abort" decision to all participants. Recall here that our CO selection and election strategies result in that the remaining final CO has access to the infrastructure (if at least one P-MN has access to the infrastructure which is a condition to initiate a meaningful generic transaction). If all P-MNs voted for committing the MT and only one CO remains for the MT, then this unique remaining CO might have a list that does not contain the IDs of all P-MNs because some votes were lost or the corresponding P-MN did not send any vote due to a transient MN failure or communication failure. In this case the expiration of the transaction lifetime will lead to a MT "Abort" decision. P-MNs share the final decisions on encounter. The final decision is inherently replicated onto the CO that turned to either a P-MN or MN-Ag since the lists of the COs are exchanged (Algorithm 11, lines 35 and 46) before electing a new CO among them. This replication is needed to recover from a failure of the last remaining CO.

Our proposed approach reduces the transaction decision time. Consequently, the resource blocking time of participants is reduced as the COs

have bounded waiting time given by the transaction lifetime for the MT outcome. If the transaction lifetime expires at one CO before reaching a final decision, the MT is aborted. This is not viable in any existing solution as P-MNs have to meet asynchronously to be able to reach a final decision or proceed with the core phase if P-FNs exist.

7.2.3 Activities of Mobile Node Agents

Upon receiving the execution fragment of its corresponding P-MN from a CO, the MN-Ag forwards it to the corresponding P-MN. After receiving E_t and S_t from the P-MN, the MN-Ag forwards this information to the CO. After receiving a "Yes" or "No" vote from the P-MN, the MN-Ag forwards the vote to the CO. Upon receiving the decision from the CO, the MN-Ag forwards it to the P-MN as soon as it is available (connected to the network). After receiving the "Ack" for decision reception from the P-MN, the MN-Ag acknowledges the CO. It is key to mention that the MN-Ag is not an active participant in the execution of the MT, since it does not have to know any information about the application and does not need to process any part or fragment of the MT.

The MN-Ag can take some decisions on behalf of its corresponding P-MN. These decisions include the extension of the timeouts of the P-MN in case of a transient disconnection. The MN-Ag is also given the responsibility to send an estimation of the timeouts of the corresponding P-MN direct after receiving the execution fragment of this P-MN (line 4). This estimation can be corrected after receiving new timeout values (E_t and S_t) from the P-MN.

7.2.4 Activities of Participant Fixed Nodes

P-FNs behave as in the established 2PC protocol, i.e., a P-FN executes its fragment, waits for the Prepare message, sends its vote and waits for the decision. Upon receiving the decision, the P-FN acknowledges the CO. Note that any existing protocol such as 3PC or Paxos Commit can be used here.

7.3 Correctness Basis

To show the correctness of the proposed PeRTAC protocol composed of Algorithm 10 , 11 , 12 and 13, we demonstrate that it satisfies the required five *atomicity properties* [Bernstein et al., 1987]:

- *Stability:* A participant cannot reverse its decision after it has reached one.

Algorithm 13: MN-Ag's Activities in PeRTAC

```
1  wait for receiving execution fragment e_i(P-MN) of the corresponding P-MN
   from CO;
2  create a Token for e_i(P-MN);
3  set the state of e_i(P-MN) to "idle" in T_i's Token;
4  send an estimation of the timeouts of corresponding P-MN to the CO;
5  for any received message do
6      if message contains the timeouts of the P-MN then
7          update T_i's Token with the received timeouts;
8          set the state of the e_i(P-MN) to "active" in T_i's Token;
9          forward the timeouts to the CO;
10     else if message contains the updates of the corresponding P-MN then
11         update the Token with the received updates;
12         send Yes vote to CO;
13     else if message contains possible disconnection of the corresponding P-MN
       and its reasons then
14         recompute the timeouts based on disconnection reasons;
15         update the token with this information;
16         send extended timeouts to the CO;
17     else if message is sent by the CO then
18         update the Token with the received message;
19         send the received message to the corresponding P-MN as soon as it is
           available;
20     else if message is sent by P-MN then
21         update the Token with the received message;
22         send the received message to CO;
23
24 end
```

- *Consistency:* All participants that reach a decision reach the same one.

- *Validity:* The "Commit" decision can only be reached if *all* participants voted "Yes".

- *Non-Triviality:* If no failure occurs and all participants voted "Yes", then the final decision should be "Commit".

- *Termination:* At any point in execution, if all existing failures are repaired and no new failures occur for sufficiently long time, then all participants will eventually reach a decision.

It follows directly from the description of the PeRTAC protocol in Section 7.2 that it satisfies the stability and the non-triviality properties. We now show that it also satisfies the consistency, validity and termination properties.

Consistency: This is satisfied as only the last active CO decides the outcome of the transaction in case the final decision is "Commit" and distributes the final decision to every participant. Hence, the last remaining CO is the single one which can have the final eventual complete list of P-MNs since at least its vote was not communicated to any other CO or P-MN according to the specification of the PeRTAC protocol. This CO is the single one able to start the core phase. If more than one CO are still remaining in the system, they can only take an "Abort" decision and no "Commit". Thus the consistency property is guaranteed by our protocol.

Validity: We assume that one of the preselected COs decides to commit the transaction when at least one participant has not decided yet. If this participant is a P-MN then its ID cannot appear in any list L (Algorithm 11, line 7) of the preselected COs according to the specification of the PeRTAC protocol. Obviously, no preselected CO can then take the decision to pre-commit the MT since this contradicts with the protocol specification (Algorithm 12, lines 1-21). In the case that at least one of the P-MNs decides to abort the transaction, the preselected COs cannot decide to pre-commit the transaction because this decision will violate the protocol specification (Algorithm 11, lines 18-21). If this participant is a P-FN, then the 2PC protocol decides to commit the transaction before receiving all the votes from the P-FNs, which again contradicts the specification of the 2PC protocol. In the case that at least one of the P-FNs decides to abort the transaction, the CO cannot decide to commit the whole transaction because this decision will also violate the 2PC protocol specification. Hence, the commit decision can only be reached if all P-MNs voted "*Yes*", i.e., decided to commit the transaction.

Termination: We consider any execution containing the failures listed in the perturbation model detailed in Section 3.3. From the PeRTAC protocol specification, we can observe that because we are using a timeout concept the protocol cannot block forever (the blocking of the protocol forever leads to a non-termination of the protocol). If at any point in execution all existing failures are repaired and no new failures occur for sufficiently long time, then all participants will eventually reach a decision. Especially in this situation all P-MNs (including COs) can meet each other eventually and progressively the lists of COs are filled and the number of COs is reduced until only one CO remains having a list L containing the IDs of all P-MNs. This complete list allows this CO to take a pre-commit decision (Algorithm 12, lines 1-21) and to start the core phase which is executed only on the wired part of the environment. The protocol terminates as soon as 2PC reaches a final decision (2PC also implements a timeout strategy to avoid blocking). If the lifetime expires at any CO before reaching the final decision, the MT is aborted

(Algorithm 11, lines 58-59) leading also to the termination of the protocol.

7.4 Performance Evaluation

We use simulations to validate our approach. We present the used performance metrics, the simulation model and our results on the high commit rate, the bounded decision time and the efficiency of PeRTAC. Our simulation studies show the feasibility of our approach and concentrate on investigating the benefit of accessing the infrastructure in MT where only MNs are participating in their execution.

7.4.1 Methodology and Simulation Settings

For the evaluation of the PeRTAC protocol, we focus on three major performance metrics: *(a) Commit rate* as it determines the service availability, *(b)* commit latency or *transaction decision time* as it determines the service response time, and *(c) message complexity* as it determines the scalability and efficiency of our approach. We measure the commit rate as the ratio of number of successfully committed MTs to total number of initiated MTs. The transaction decision time is the time needed to take a decision about the outcome of the initiated MT, i.e., the time between the initiation of the MT and the time where the final decision is reached at the CO. The blocking time of P-MNs is majorally determined by the transaction decision time. However, the time needed for the final decision to reach the P-MNs plays a role especially in mobile ad-hoc and mobile generic environments (blocking time = decision time + time needed to disseminate the final decision). This time is dependent on the implementation of the message dissemination protocols used to disseminate the final decision and therefore will not be further investigated in our performance evaluation. The message complexity of PeRTAC is defined as the number of wireless messages sent and received in average by each P-MN during the execution of the MT.

The performance of the PeRTAC approach is evaluated in this chapter based on the service delivery level assured by the protocol and defined basically by the commit rate and the decision time as described in Chapter 6. The costs of assuring a certain service delivery level are measured in terms of message complexity. We focus in our performance evaluation on the impact of BS coverage and network partitioning degree of MNs that can only communicate in ad-hoc manner on the identified performance metrics.

For our simulation studies we have used J-Sim [J-Sim], a component-based, compositional simulation environment that is entirely developed in

Java (see [Tyan et al., 2009] for further details of the simulation environment). For the performance evaluation of the PeRTAC protocol, we consider a representative range of parameter values to assess the described approach. Table 7.2 summarizes our simulation settings. We selected the commonly used Random Waypoint mobility model [Broch et al., 1998] (node speed uniform in [0.5, 1.5] m/s). We fix the mobility area (2km x 2km) and the communication range (250m). We generate the mobility scenarios using the BonnMotion mobility simulator [BonnMotion]. Given its importance, for all our simulation studies we vary the partitioning degree through varying the number of nodes. The partitioning degree or degree of separation is provided by BonnMotion and reflects how likely it is that two randomly chosen nodes are not within the same partition at a randomly chosen point in time.

Table 7.2: Simulation settings

Parameter	**Value(s)**
Geographical area	$2km$ x $2km$
Communication range	$250m$
Mobility models	Random Waypoint (RWP)
Node speed	uniform in [0.5, 1.5] m/s
#Nodes	$\in$ [20,200]
#COs	$\in \{3,5,7,10\}$
#P-MNs	10
lifetime	$\in \{60s,120s,300s\}$

BSs have in our simulations the same communication range as the MNs. We place the BSs uniformly in the simulated area and vary the number of deployed BSs in order to vary the coverage area of these BSs. Table 7.3 gives for every number of deployed BSs the percentage of the simulated area covered by these BSs. We consider that all deployed MNs in the simulation area can communicate with the BSs and also with other MNs (in ad-hoc mode).

We generate transactions of similar length and with execution fragments of similar length also. We initiate one transaction at the beginning of each simulation. We fix the number of P-MNs to 10 and vary the number of preselected COs, the lifetime and the number of BSs to study the impact of these parameters on the performance of PeRTAC. Each simulation is repeated 200 times for statistical significance of the results.

Table 7.3: Base station coverage in the simulated area

Number of base stations	Coverage in %
4	19.63
9	44.17
16	78,53
25	98,2
36	100

7.4.2 Simulation Results

Now, we present the results of our conducted simulation studies for the defined performance metrics. As mentioned before, we simulate PeRTAC under different network conditions and vary the important protocol parameters to study the behavior of our protocol in a wide range of possible deployment scenarios. Overall, we split the results for "Abort" and "Commit" cases to have better insights to PeRTAC.

Impact of BSs' Coverage

We arbitrarily fix in this scenario the number of preselected COs to 3 and the transaction lifetime value to 300 s. To assess the influence of the BSs' coverage area on the PeRTAC protocol, we vary the number of the BSs deployed in the simulation area (i.e., we vary the percentage of the simulated area covered by these BSs).

Figure 7.5 shows that PeRTAC benefits from accessing BSs to increase the number of committed transactions compared to ParTAC (where the number of BSs is 0). Accessing the infrastructure enables some partitions to communicate with other partitions through the infrastructure despite the fact that the MNs in these partitions are not able to communicate with each other in ad-hoc multihop manner.

In Figure 7.6 we observe also that accessing the infrastructure reduces the decision time if the number of mobile nodes in the area is relatively small (less than 100 MNs in our simulations). The reduction of the decision time can be explained by the fact that partitions which are connected through the infrastructure do not need to wait until they merge to make a progress like it is the case in ParTAC. If the number of MNs deployed in the simulation area increases (greater than 100), the decision time increases as the BSs become a bottleneck in this case because the number of the MNs in their coverage

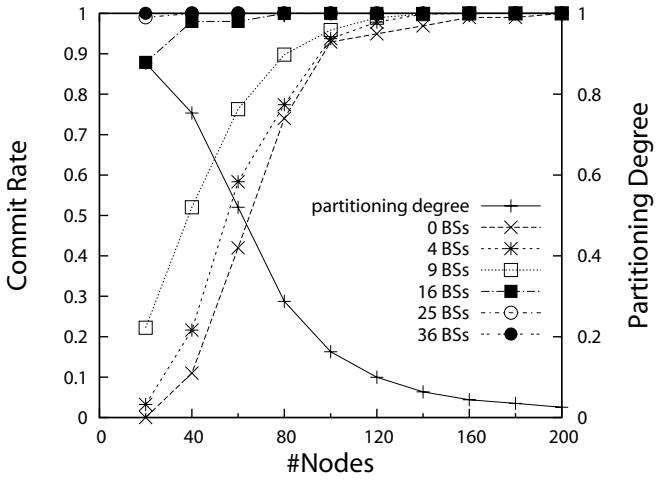

Figure 7.5: Impact of BSs' coverage on commit rate

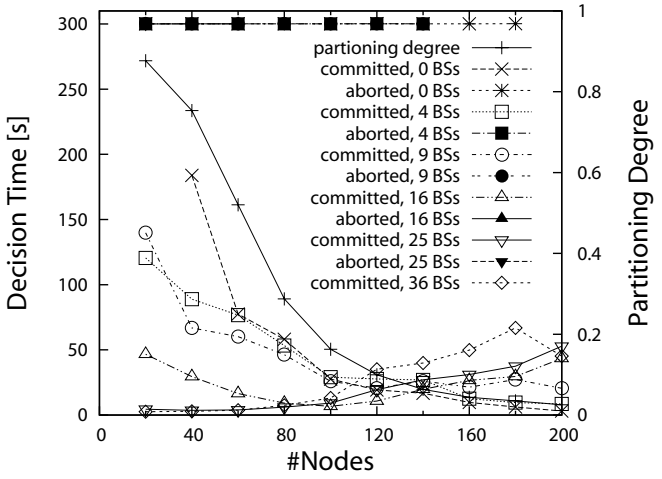

Figure 7.6: Impact of BSs' coverage on decision time

area also increases (recall that we have chosen a relatively low communication range for the BSs in our simulations). Therefore the throughput of these BSs decreases and more time is needed to execute the PeRTAC protocol. Apart from some "Abort" cases where the partitioning degree is very high and consequently more message are exchanged, Figure 7.7 shows that the costs of the MTs in term of exchanged wireless messages remain almost constant in our simulations.

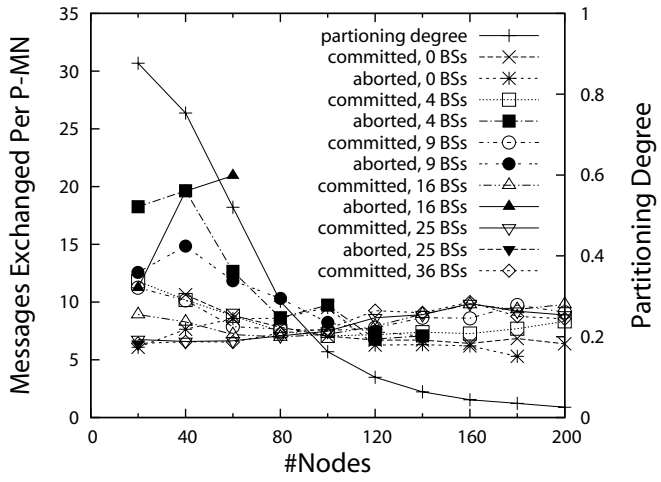

Figure 7.7: Impact of BSs' coverage on message complexity

Impact of Transaction Lifetime

We arbitrarily fix in this scenario the number of preselected COs to 3 and number of deployed BSs to 16 (i.e., 78.53% of the simulated area is covered by the BSs). To assess the influence of the lifetime on the PeRTAC protocol, we select the following lifetime values: 60 s, 120 s and 300 s.

Figure 7.8 shows that also in the case of PeRTAC, the transaction commit rate depends on the lifetime of the initiated MT. If we increase the transaction lifetime, the commit rate increases. If the number of deployed MNs in the simulated area is greater than 100, the commit rate starts even to decrease because the decision time starts to increase as illustrated in Figure 7.9 and as explained above when the impact of the BSs' coverage is investigated.

In Figure 7.10, we observe that the number of exchanged messages in the "Abort" case increases if the lifetime increases since an increase in the lifetime implies an increase in the number of exchanged messages (more votes and lists can be sent to encountered COs if lifetime increases). For the "Commit" case the message complexity does not change if the lifetime is changed.

Figures 7.9 and 7.10 show for the "Aborted" case where the lifetime is 120 s an irregular behavior when the number of nodes is 100. This is due to the fact that in the point the commit rate as shown in Figure 7.8 is 100% and therefore the number of aborted transaction is 0, which translates in 0 s decision time and 0 exchanged messages per P-MN.

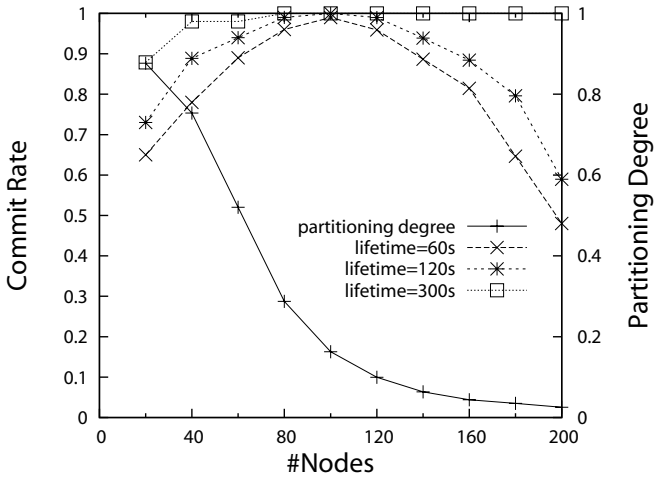

Figure 7.8: Impact of lifetime on commit rate

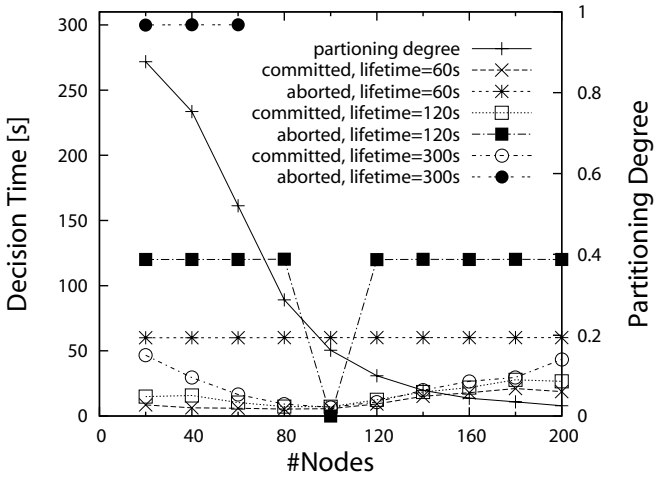

Figure 7.9: Impact of lifetime on decision time

Impact of Number of Preselected COs

We arbitrarily fix in this scenario the lifetime to 300 s and the number of deployed BSs to 9 (i.e., 44.17% of the simulated area is covered by the BSs) and vary the number of preselected COs.

The number of preselected COs does not impact the commit rate of PeR-TAC as illustrated in Figure 7.11. This is due to the fact that as soon as two COs encounter each other only one of them remains active and the other one

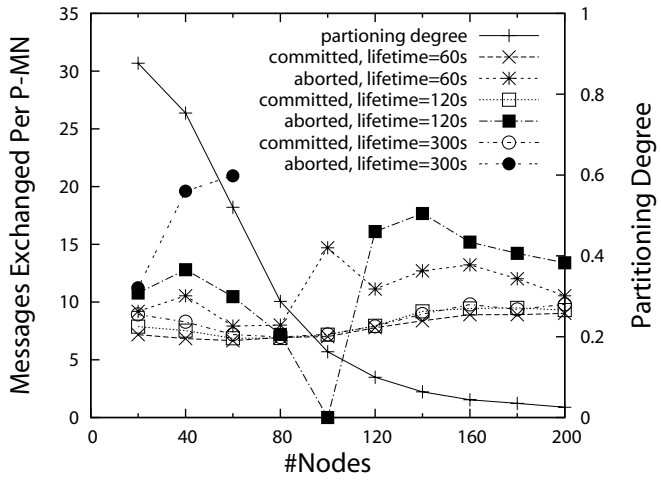

Figure 7.10: Impact of lifetime on message complexity

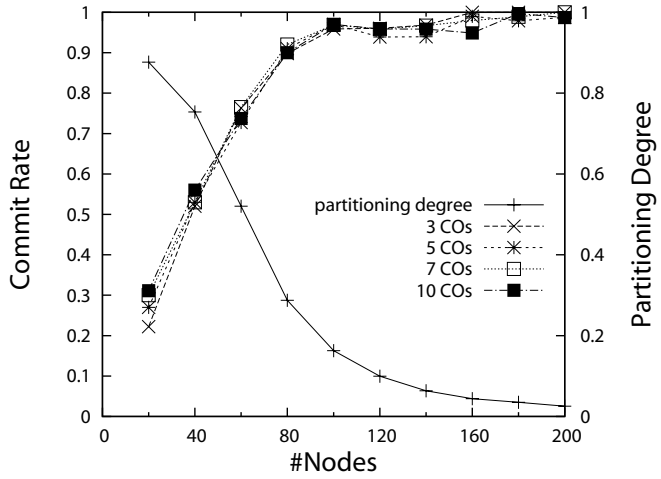

Figure 7.11: Impact of number of COs on commit rate

becomes a normal P-MN. After a certain point in time only a few (2 to 3) COs remain and all the simulated scenarios behave from this instant onwards similarly. This point in time is closer to the initiation time of the MT in the "Commit" case as from all the COs present in one partition only one remains active as soon as they receive beacons from each other. Figure 7.12 shows that the number of preselected COs does have only a slight impact on the decision time also because of the same reasons given above.

However, the number of preselected COs has a minor impact on the effi-

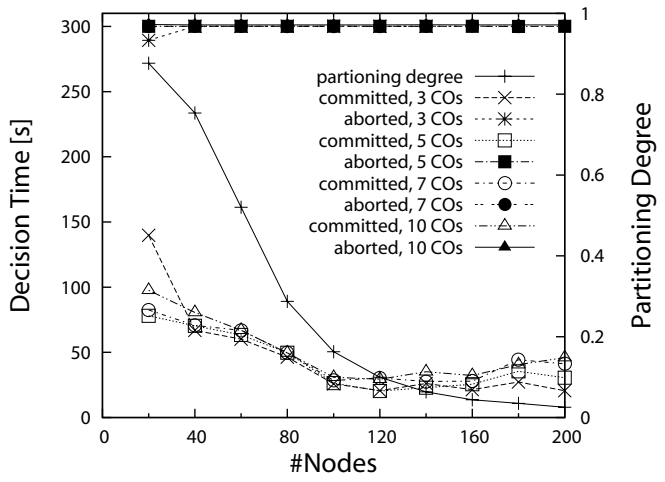

Figure 7.12: Impact of number of COs on decision time

ciency of the PeRTAC protocol as shown in Figure 7.13. The slight increase of the number of messages exchanged per node is due to the fact that every P-MN needs to send its vote to more COs as the number of COs increases. It is noteworthy to mention that selecting higher number COs is primarily to tolerate CO failures during the MT execution. Our simulations show that a higher CO failure-tolerance does only slightly impact the message efficiency.

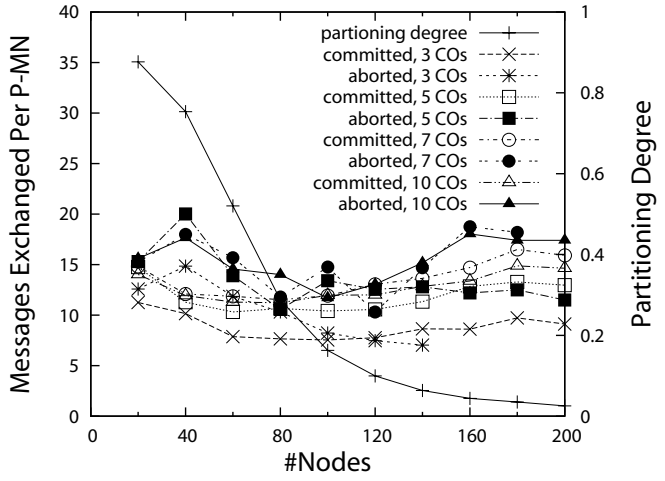

Figure 7.13: Impact of number of COs on message complexity

7.4.3 Discussion

Our simulation studies in this chapter show that the PeRTAC approach takes advantage of the access to the infrastructure whenever possible to achieve better performance especially with respect to the transactional service availability by increasing the commit rate of initiated mobile transactions and with respect to commit latency by reducing the commit decision time of these transactions. Similar to FT-PPTC and ParTAC, the PeRTAC commit protocol takes also into consideration application requirements by defining an appropriate transaction lifetime for each initiated mobile transaction. We have shown through our evaluation studies the existence of a trade-off between the chosen lifetime and the performance of the PeRTAC protocol in terms of commit rate, decision time and message complexity (the same trade-off was observed in the evaluation of the FT-PPTC and ParTAC approaches in Chapter 5 and Chapter 6 respectively). By defining an appropriate lifetime, the application also limits and controls the cost of the initiated transactions.

7.5 Chapter Summary

As each evolving mobile environment necessitates new commit constraints, the current approaches geared towards dedicated scenarios, often do not provide comprehensive and generic commit capabilities. Thus we developed in this chapter a generic and evolvable atomic commit solution that benefits from the presence of an infrastructure and delivers best-effort results in its absence. We have introduced the main challenges for designing atomic commit protocols faced in generic mobile environments. We presented PeRTAC, an efficient perturbation-resilient commit protocol that provides strict atomicity in spite of frequent mobile environment perturbations. Especially PeRTAC fills the gap between solutions provided for mobile infrastructure-based and mobile ad-hoc (i.e., infrastructure-less) environments.

Chapter 8

Conclusions and Future Research

In this thesis we have investigated atomic commit protocols in mobile environments. The existing atomic commit protocols are appropriate for some application scenarios, however, they show significant drawbacks in perturbation-prone scenarios. In this thesis we developed a perturbation-resilient framework for atomic commit protocols in mobile environments.

This chapter concludes the thesis by summarizing our main contributions and discussing their extendability. In this light, we sketch possible extensions of our framework to consider other dependability aspects like security and to deploy the developed framework of atomic commit protocols in real world scenarios. We believe that the work presented in this thesis opens up new interesting research directions.

8.1 Overall Thesis Contributions

The main goal of the thesis was to study perturbation-resilient atomic commit protocols in mobile environments. Our research effort was driven by the current need for perturbation-resilient and efficient solutions in the studied mobile environment where perturbations are part of the normal behavior of the environment and not an exception. Accordingly, this section discusses the key contributions made by the research presented in this thesis. Driven by the research problem introduced in Section 1.1 and grouped by topic, the thesis contributions are surveyed and their relevance is discussed.

8.1.1 Investigation of Perturbations in the Mobile Environment

In order to build the background of our study of perturbation-resilient atomic commit protocols in mobile environments, a perturbation-resilient framework was developed in this work to provide strict atomicity for transactions in mobile environments. The main goal of this framework was to investigate all the perturbations in mobile environments that can affect and disturb the normal execution of atomic commit protocols in such perturbation-prone environments. This study builds the background and facilitates the deployment of a wide range of application scenarios in mobile environments where strict atomicity is a requirement. To build our framework, we studied for every identified perturbation the mechanisms and techniques used to tolerate it. The identified and presented mechanisms and techniques represent the building blocks of our modular framework of perturbation-resilient transaction atomic commit protocols for mobile infrastructure-based, mobile ad-hoc and mobile generic environments.

8.1.2 A Modular Framework of Perturbation-Resilient Transaction Atomic Commit Protocols

The next step in this thesis was building a modular framework of perturbation-resilient atomic commit protocols in the different identified types of mobile environments, i.e. infrastructure-based, ad-hoc and generic environments.

Mobile Infrastructure-based Environment: We started building our modular framework for perturbation-resilient atomic commit protocols in

mobile environments by providing a family of progressive mobile transaction commit protocols (PPTC, FT-PPTC, FT-PPTC-Rec) for three common classes of mobile infrastructure-based environments. This family of solutions is extendable and adaptable in its nature which allows the designer to customize protocol resilience to a subset of all possible perturbations based on the overall costs involved such as message complexity and costs for bandwidth utilization.

We have also demonstrated, using simulations and real experiments, the efficiency, scalability and level of perturbation-resilience added by using our proposed perturbation-tolerance techniques. We showed in our studies the utility of the different building blocks we developed in our framework to cope with perturbations specific to the mobile infrastructure-based environment. We demonstrated how these blocks can be combined during design time to adapt the atomic commit protocol to the characteristics and perturbations of the considered mobile environment. Overall, the family of protocols developed for mobile infrastructure-based environments outperforms the existing solutions with respect to efficiency and fault-tolerance. This is achieved by adding a tolerable overhead in terms of message complexity and transaction execution times.

Mobile Ad-Hoc Environment: Next, we extended our modular framework for perturbation-resilient atomic commit protocols in mobile environments by introducing ParTAC, a novel atomic transaction commit protocol that provides strict atomicity in mobile ad-hoc environments in spite of frequent perturbations in this environment and especially network partitioning. ParTAC is independent from the considered mobile ad-hoc environment and it is generalized since it is not based on hard assumptions like consensus and group membership. Being atomic and efficient, and maximizing the commit rate, ParTAC guarantees data consistency while allowing for high transactional service availability and scalability.

Transactional services represent a key part of service oriented architectures and increasingly for mobile ad-hoc environments, vehicular ad-hoc networks etc. ParTAC guarantees consistency of data and maximizes the commit rate. This is achieved for mobile ad-hoc environments showing an arbitrary degree of perturbations with respect to network partitioning as illustrated in our performance evaluation studies. Furthermore, the ParTAC approach helps in reducing the transaction decision time resulting in a better transactional throughput and consequently in a better scalability. This allows maximizing the number of users that can use the database resources on resource-limited mobile nodes.

Mobile Generic Environment: Finally, the modular framework for perturbation-resilient atomic commit protocols in mobile environments was augmented by the PeRTAC protocol that represents a generic and evolvable atomic commit solution in mobile generic environments. PeRTAC benefits from the presence of an infrastructure and delivers best-effort results in its absence. PeRTAC is an efficient perturbation-resilient commit protocol that provides strict atomicity in spite of frequent mobile environment perturbations. Especially, PeRTAC fills the gap between solutions provided for mobile infrastructure-based and mobile ad-hoc (i.e., infrastructure-less) environments.

The performance evaluation of PeRTAC shows that this approach takes advantage of accessing the wired infrastructure whenever possible to achieve better performance especially: *(i)* with respect to the transactional service availability by increasing the commit rate of initiated mobile transactions and *(ii)* with respect to commit latency by reducing the commit decision time of these transactions. Furthermore, our evaluation studies have shown the existence of a trade-off between the chosen lifetime and the performance of the PeRTAC protocol in terms of commit rate, decision time and message complexity. By defining an appropriate lifetime, the application also limits and controls the cost of the initiated transactions.

8.2 Application Scenario Implementation

Despite the fact that almost all protocols described in this thesis where implemented on mobile devices using the J2ME (Java 2 Micro Edition) [Java ME], we targeted also the implementation of a complete application scenario to show how the work done in this thesis can be useful in real life. We have chosen the coordination across autonomous vehicles application scenario described in Section 4.1. We describe now briefly the application and the corresponding real world implementation.

8.2.1 Application Scenario Description

Data-based agreement is increasingly used to implement traceable coordination across mobile entities such as ad-hoc networked (autonomous) vehicles. In our implementation, we focused on data-based agreement using database transactions where mobile entities agree on a set of coordinated tasks that need to be performed by them in an atomic way. The data about the agreed tasks and their corresponding stakeholders are kept in local databases as a proof for the obtained agreement. This proof might be needed by users

and regularities/authorities involved depending on the application scenario. Through this application scenario we demonstrate our effort to provide for partition-aware atomic commit protocols for transactional data-based agreement.

Coordination across autonomous networked vehicles is an excellent scenario for atomic transaction protocols in mobile ad-hoc environments. It presents a potential application where mobile transactions are needed for the purpose of coordination for safe and traceable navigation of unmanned autonomous networked vehicles. Like airplanes, we assume in our application scenario that autonomous vehicles are equipped with black boxes which are basically mobile databases. Figure 4.1 shows four unmanned vehicles at a traffic intersection. These vehicles need to agree on an order how they will pass the intersection. Prior to their actual passing, this order information needs to be agreed upon and recorded atomically to their corresponding black boxes. This information would be absolutely useful for insurance companies or the police department in case an accident occurs between these vehicles.

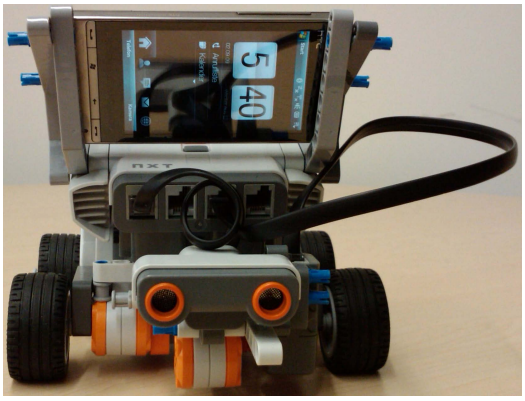

Figure 8.1: LEGO Mindstorms equipped with HTC PDA

8.2.2 Prototype Implementation

To demonstrate the inter-vehicle scenarios described above, we use four LEGO Mindstorms NXT 2.0 robots [Lego Mindstorms] equipped with ultrasonic and color sensor to play the role of the unmanned vehicles. We select the HTC Touch Diamond 2 [HTC Touch Diamond 2] Personal Digital Assistant (PDA) as our development platform as it provides a WiFi interface, large memory and a good development environment. Every robot is equipped with one PDA (Figure 8.1) that represents the computation and communication unit of the unmanned vehicle. The PDA sends commands

to the robot via Bluetooth. These commands control the movement of the robot and sensing activity of its environment.

The unmanned vehicles communicate with each other using WiFi (IEEE 802.11b/g) interfaces of their corresponding computation and communication units (PDAs). We implement our application using J2ME (Java 2 Micro Edition). For this purpose, we install on every PDA a J2ME virtual machine (IBM J9 MIDP 2.0 [IBM J9]). We install on every PDA a mobile database (Mimer Mobile SQL [Mimer SQL Mobile]) needed for the realization of the black boxes described in the inter-vehicle coordination scenarios in Section 8.2.1. We implement the ParTAC protocol described in Chapter 6 to coordinate the unmanned vehicles passage at the traffic intersection. Using their sensors, the robots are able to detect their position in the field and the information is transmitted to their corresponding PDAs via Bluetooth. Every vehicle in our demonstration has a unique ID.

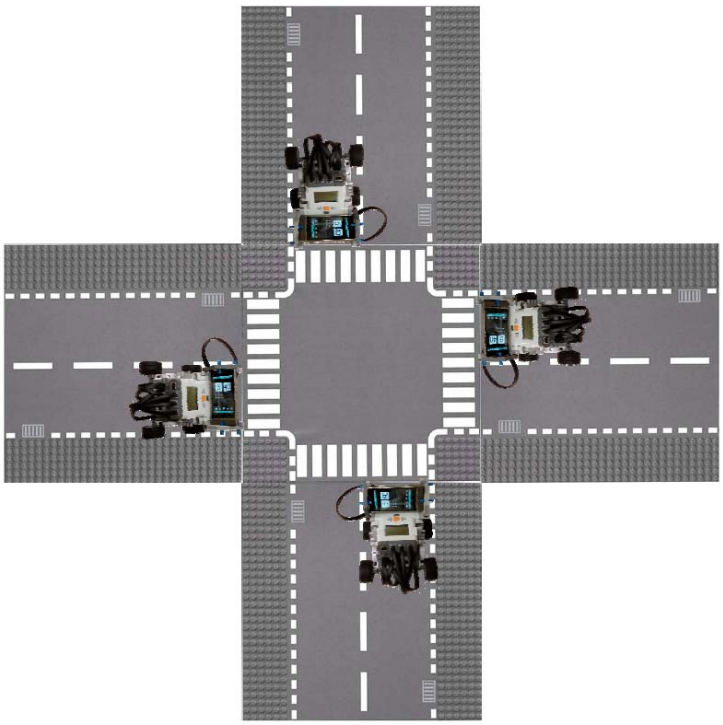

Figure 8.2: Intersection scenario

8.2.3 Demonstration

We demonstrated our application scenario in the 11th International Conference on Mobile Data Management (MDM 2010), held in the city of Kansas

City, Missouri, from May 23rd to May 26th, 2010. For our demonstration we placed the vehicles (LEGO Mindstorms NXT 2.0 equipped with a PDA each) on a virtual traffic intersection as shown in Figure 8.2. Using their sensors the vehicles are able to detect an intersection. The first vehicle which reaches the intersection will stop and detect the presence of other vehicles in the intersection by broadcasting a HELLO message. Every vehicle that receives this message, responds with its ID. Upon receiving the reply messages, the initiator vehicle sends a transaction to all the other vehicles present in the intersection. The order of passage of the vehicles is determined by the order of the messages received from the other vehicles present in the intersection. Using the ParTAC protocol, the application decides either to commit or abort the transaction. In case the transaction commits, the intersection passage order of the unmanned vehicles is saved in the database and send to all other vehicles.

8.3 Open Ends - Basis for Future Work

While the work presented in this thesis addressed the research problem driving it towards making the discussed contributions, it also opened new and interesting research perspectives along its way. In the following, we briefly present some of the most promising ones.

- We studied in this thesis atomic commit protocols in mobile environments which are deployed mainly to guarantee the Atomicity and Consistency transaction properties (we refer to the transaction ACID properties defined in Chapter 1). The framework developed in this thesis can be extended to investigate concurrency control protocols (which ensure the Isolation property), recovery and replication protocols (to guarantee the Durability property).

- Another important issue that is gaining importance in mobile environments is security. In this thesis, we assumed all entities to be trusted. Eventually, the design of atomic commit protocols will also have to incorporate concepts that protect against malicious nodes. One possible research direction in this field is to study the behavior of mobile transaction users and store this behavior in user profiles that can be used in future to detect abnormal and eventually malicious behavior of nodes participating in the mobile transaction execution.

- The multiple coordinator strategy used in the mobile ad-hoc and generic environments showed its strengths in providing partition-tolerant solutions for the atomic transaction commit problem in

infrastructure-less scenarios. In our evaluation studies, we adopted a scheme of selecting randomly the preselected COs from the set of P-MNs. It will be interesting to study other schemes such selecting the preselected COs based on their connectivity, computational and storage capabilities and mobility patterns compared to others P-MNs and to identify some heuristics which define how this selection should be performed.

- Another interesting extension of this work is to investigate the atomic transaction commit problem in wireless sensor networks which represent an emerging and promising research field. Wireless sensor networks can be seen as a special case of mobile ad-hoc environments where the nodes (sensor nodes) are significantly constrained in computational, storage and energy resources compared to laptops, PDAs and mobile phones.

- Finally, the extension of the implementation of the protocols performed during this work to other real-world deployments. Especially it is interesting to develop a testing framework to assess the perturbation-resilience of the developed approaches in these real-world deployments.

Bibliography

Emile Aarts, Rick Harwig, and Martin Schuurmans. Ambient intelligence. In *The Invisible Future: The Seamless Integration Of Technology Into Everyday Life*. McGraw-Hill Companies, 2001.

Rafael Alonso and Henry F. Korth. Database system issues in nomadic computing. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 388–392, 1993.

Philip A. Bernstein, Vassos Hadzilacos, and Nathan Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, 1987.

Christophe Bobineau, Philippe Pucheral, and Maha Abdallah. A unilateral commit protocol for mobile and disconnected computing. In *Proceedings of the 12th International Conference on Parallel and Distributed Computing and Systems (PDCS)*, 2000.

Christophe Bobineau, Cyril Labbé, Claudia Lucia Roncancio, and Patricia Serrano-Alvarado. Comparing transaction commit protocols for mobile environments. In *Proceedings of the 15th International Workshop on Database and Expert Systems Applications*, pages 673–677, 2004.

BonnMotion. BonnMotion: The BonnMotion Mobilty Scenario Generation and Analysis Tool. http://iv.cs.uni-bonn.de/wg/cs/applications/bonnmotion/, 2009.

Joos-Hendrik Bose, Stefan Böttcher, Le Gruenwald, Sebastian Obermeier, Heinz Schweppe, and Thorsten Steenweg. An integrated commit protocol for mobile network databases. In *Proceedings of the 9th International Database Engineering & Application Symposium (IDEAS)*, pages 244–250, 2005.

Stefan Böttcher, Le Gruenwald, and Sebastian Obermeier. A failure tolerating atomic commit protocol for mobile environments. In *Proceedings of the*

8th International Conference on Mobile Data Management (MDM), pages 158–165, 2007.

Yuri Breitbart, Hï¿½ctor Garcï¿½a-Molina, and Abraham Silberschatz. Overview of multidatabase transaction management. *VLDB Journal*, 1 (2):181–239, 1992.

Linda Briesemeister and Günter Hommel. Localized group membership service for ad hoc networks. In *Proceedings of the International Workshop on Ad Hoc Networking (IWAHN)*, pages 94–100, 2002.

Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking (MobiCom)*, pages 85–97, 1998.

Ernest Chang and Rosemary Roberts. An improved algorithm for decentralized extrema-finding in circular configurations of processes. *Communications of the ACM*, 22(5):281–283, 1979.

Panos K. Chrysanthis. Transaction processing in a mobile computing environment. In *Proceedings of the IEEE Workshop on Advances in Parallel and Distributed Systems (APADS)*, pages 77–82, 1993.

Susan B. Davidson, Hector Garcia-Molina, and Dale Skeen. Consistency in a partitioned network: a survey. *ACM Computing Surveys*, 17(3):341–370, 1985.

Margaret H. Dunham, Abdelsalam Helal, and Santosh Balakrishnan. A mobile transaction model that captures both the data and movement behavior. *Mobile Networks and Applications*, 2(2):149–162, 1997.

E. N. (Mootaz) Elnozahy, Lorenzo Alvisi, Yi-Min Wang, and David B. Johnson. A survey of rollback-recovery protocols in message-passing systems. *ACM Computing Surveys*, 34(3):375–408, 2002.

George H. Forman and John Zahorjan. The challenges of mobile computing. *IEEE Computer*, 27(4):38–47, 1994.

Hï¿½ctor Garcï¿½a-Molina. Elections in a distributed computing system. *IEEE Transactions on Computers*, 31(1):48–59, 1982.

Hï¿½ctor Garcï¿½a-Molina. Using semantic knowledge for transaction processing in a distributed database. *ACM Transactions on Database Systems*, 8(2):186–213, 1983.

Gartner. Gartner Consulting Firm. http://www.fortune500global.com/news/number-of-computers-sold-worldwide-in-2009-rose-gartner/, 2009.

Jim Gray. Notes on data base operating systems. In *Operating Systems, An Advanced Course*, pages 393–481, January 1978.

Jim Gray and Leslie Lamport. Consensus on transaction commit. *ACM Transactions on Database Systems*, 31(1):133–160, March 2006.

Jim Gray, Pat Helland, Patrick O'Neil, and Dennis Shasha. The dangers of replication and a solution. pages 372–381, 1998.

Jörg Hähner, Dominique Dudkowski, Pedro José Marrón, and Kurt Rothermel. A quantitative analysis of partitioning in mobile ad hoc networks. *ACM SIGMETRICS Performance Evaluation Review*, 32(1):400–401, 2004.

Jörg Hähner, Dominique Dudkowski, Pedro José Marrón, and Kurt Rothermel. Quantifying network partitioning in mobile ad hoc networks. In *Proceedings of the 8th International Conference on Mobile Data Management (MDM)*, pages 174–181, 2007.

Theo Härder and Andreas Reuter. Principles of transaction-oriented database recovery. *ACM Computing Surveys*, 15(4):287–317, 1983.

Theo Härder and Andreas Reuter. *Principles of transaction-oriented database recovery*. Morgan Kaufmann Publishers Inc., 1994.

Xiaoyan Hong, Mario Gerla, Guangyu Pei, and Ching-Chuan Chiang. A group mobility model for ad hoc wireless networks. In *Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems (MSWiM)*, pages 53–60, 1999.

HTC Touch Diamond 2. HTC Touch Diamond 2: Personal Digital Assistant. http://www.htc.com/de/product/touchdiamond2/overview.html, 2010.

Elgan Huang, Wenjun Hu, Jon Crowcroft, and Ian Wassell. Towards commercial mobile ad hoc network applications: A radio dispatch system. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, pages 355–365, 2005.

IBM DB2. IBM Software Products, DB2 Product Family. http://www-01.ibm.com/software/data/db2/, 2009.

IBM DB2 Everyplace. IBM Software Products, DB2 Everyplace. http://www-01.ibm.com/software/data/db2/everyplace/, 2009.

IBM J9. IBM J9: IBM Java Virtual Machine. http://www-01.ibm.com/software/wireless/weme/, 2010.

Tomasz Imielinski and B. R. Badrinath. Mobile wireless computing: challenges in data management. *Communications of the ACM*, 37(10):18–28, 1994.

Information Week, 2009. Information Week business technology magazine. http://www.informationweek.com/news/internet/webdev/showArticle.jhtml?articleID=222001329, 2009.

J-Sim. J-Sim: The J-Sim Website. http://sites.google.com/site/jsimofficial/, 2005.

J-Sim Wireless. J-Sim Wireless: The J-Sim Wireless Extension Tutorial. http://www.j-sim.org/v1.3/wireless/wireless_tutorial.htm, 2005.

Java ME. Java ME: Java 2 Micro Edition. http://java.sun.com/javame/, 2010.

Randi Karlsen. An adaptive transactional system - framework and service synchronization. In *Proceedings of the 5th International Symposium on Distributed Objects and Applications (DOA)*, pages 1208–1225, 2003.

Abdelmajid Khelil, Pedro José Marrón, Christian Becker, and Kurt Rothermel. Hypergossiping: A generalized broadcast strategy for mobile ad hoc networks. *Ad Hoc Networks*, 5(5):531–546, 2007.

Abdelmajid Khelil, Faisal Karim Shaikh, Brahim Ayari, and Neeraj Suri. MWM: a map-based world model for event-driven wireless sensor networks. In *Proceedings of the 2nd International Conference on Autonomic Computing and Communication Systems (AUTONOMICS)*, pages 1–10, 2008.

Abdelmajid Khelil, Faisal Karim Shaikh, Piotr Szczytowski, Brahim Ayari, and Neeraj Suri. Map-based design for autonomic wireless sensor networks. In *Autonomic Communication*, pages 309–326. Springer US, 2009.

Abdelmajid Khelil, Christian Reinl, Brahim Ayari, Faisal Karim Shaikh, Piotr Szczytowski, Azad Ali, and Neeraj Suri. Sensor cooperation for a sustainable quality of information. In *Pervasive Computing and Networking*. John Wiley, (accepted, to appear), 2010.

James J. Kistler and Mahadev Satyanarayanan. Disconnected operation in the coda file system. *ACM Transactions on Computer Systems*, 10(1): 3–25, 1992.

Kyong-I Ku and Yoo-Sung Kim. Moflex transaction model for mobile heterogeneous multidatabase systems. In *Proceedings of the 10th International Workshop on Research Issues in Data Engineering (RIDE)*, page 39, 2000.

Vijay Kumar. A timeout-based mobile transaction commitment protocol. In *Proceedings of the East-European Conference on Advances in Databases and Information Systems*, pages 339–345, 2000. ISBN 3-540-67977-4.

Vijay Kumar and Margaret H. Dunham. Defining location data dependency, transaction mobility and commitment. TR 98-CSE-1, Southern Methodist Univ., February 1998.

Vijay Kumar, Nitin Prabhu, Magaret H. Dunham, and Ayse Yasemin Seydim. TCOT-A timeout-based mobile transaction commitment protocol. *IEEE Transactions on Computers*, 51(10):1212–1218, 2002.

Leslie Lamport. The part-time parliament. *ACM Transactions on Computer Systems*, 16(2):133–169, 1998.

Leslie Lamport and Mike Massa. Cheap paxos. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, page 307, 2004.

Lego Mindstorms. LEGO Mindstorms. LEGO Website. http://mindstorms.lego.com, 2010.

Sanjay Kumar Madria and Bharat Bhargava. A transaction model to improve data availability in mobile computing. *Distributed Parallel Databases*, 10 (2):127–160, 2001.

Navneet Malpani, Jennifer L. Welch, and Nitin Vaidya. Leader election algorithms for mobile ad hoc networks. In *Proceedings of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications (DIALM)*, pages 96–103, 2000.

Salahuddin Mohammad Masum, Amin Ahsan Ali, and Mohammad Touhid youl Islam Bhuiyan. Asynchronous leader election in mobile ad hoc networks. In *Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA)*, pages 827–831, 2006.

Mimer SQL Mobile. Mimer SQL Mobile: Mimer Relational DBMS for mobile devices. http://www.mimer.com/leftright.asp?secId=172, 2010.

C. Mohan and B. Lindsay. Efficient commit protocols for the tree of processes model of distributed transactions. *ACM SIGOPS Operating Systems Review*, 19(2):40–52, 1985.

Nadia Nouali, Anne Doucet, and Habiba Drias. A two-phase commit protocol for mobile wireless environment. In *Proceedings of the 16th Australasian Database Conference (ADC)*, pages 135–143, 2005.

Nadia Nouali-Taboudjemat and Habiba Drias. A policy-based context-aware approach for the commitment of mobile transactions. In *Proceedings of the 8th international conference on New technologies in distributed systems (NOTERE)*, pages 1–11, 2008.

Nadia Nouali-Taboudjemat, Lynda Boukantar, and Habiba Drias. Performance evaluation of atomic commit protocols for mobile transactions. *International Journal of Intelligent Information and Database Systems*, 1(2): 122–155, 2007.

Sebastian Obermeier, Stefan Böttcher, and Dominik Kleine. CLCP-A distributed cross-layer commit protocol for mobile ad hoc networks. In *Proceedings of the 6th International Symposium on Parallel and Distributed Processing with Applications (ISPA)*, pages 361–370, 2008.

Sebastian Obermeier, Stefan Böttcher, Martin Hett, Panos K. Chrysanthis, and George Samaras. Blocking reduction for distributed transaction processing within manets. *Distributed Parallel Databases*, 25(3):165–192, 2009.

Oracle Database Lite. Oracle Corporation, Oracle Database Lite 10g: The Internet Platform For Mobile Computing. http://www.oracle.com/technology/products/lite/index.html, 2009.

Oracle Database Standard Edition. Oracle Corporation, Oracle Database 11g Standard Edition. http://www.oracle.com/database/standard_edition.html, 2009.

Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *ACM SIGCOMM Computer Communication Review*, 24(4):234–244, 1994.

Charles E. Perkins and Elizabeth M. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 90–100, 1999.

Evaggelia Pitoura and Bharat Bhargava. Revising transaction concepts for mobile environments. In *Proceedings of the 1st IEEE Workshop on Mobile Computing Systems and Applications*, pages 164–168, 1994.

Evaggelia Pitoura and Bharat Bhargava. Maintaining consistency of data in mobile distributed environments. In *Proceedings 15th International Conference on Distributed Computing Systems (ICDCS)*, pages 404–413, 1995.

Evaggelia Pitoura and Bharat Bhargava. Data consistency in intermittently connected distributed systems. *IEEE Transactions on Knowledge and Data Engineering*, 11(6):896–915, 1999.

Dhiraj K. Pradhan, P. Krishna, and Nitin H. Vaidya. Recovery in mobile wireless environment: Design and trade-off analysis. In *Proceedings of the 26th International Symposium on Fault-Tolerant Computing (FTCS)*, pages 16–25, 1996.

Gruia-Catalin Roman, Qingfeng Huang, and Ali Hazemi. Consistent group membership in ad hoc networks. In *Proceedings of the 23rd International Conference on Software Engineering (ICSE)*, pages 381–388, 2001.

Patricia Serrano-Alvarado. *Transactions Adaptables pour les Environnements Mobiles*. PhD thesis, Joseph-Fourier University, Grenoble, France, 2004.

Patricia Serrano-Alvarado, Claudia Roncancio, Michel Adiba, and Cyril Labbé. Adaptable mobile transactions and environment awareness. In *Proceedings of the 19ï¿½mes Journï¿½es Bases de Donnï¿½es Avanceï¿½s (BDA)*, 2003.

Patricia Serrano-Alvarado, Claudia Roncancio, and Michel Adiba. A survey of mobile transactions. *Distributed Parallel Databases*, 16(2):193–230, 2004a.

Patricia Serrano-Alvarado, Claudia Roncancio, Michel Adiba, and Cyril Labbï¿½. Context aware mobile transactions. In *Proceedings of the 5th International Conference on Mobile Data Management (MDM)*, page 167, 2004b.

Patricia Serrano-Alvarado, Claudia Lucia Roncancio, Michel Adiba, and Cyril Labbé. An Adaptable Mobile Transaction Model for Mobile Environments. *International Journal Computer Systems Science and Engineering – Special issue on Mobile Databases, 20, 3*, 2005.

Faisal Karim Shaikh, Abdelmajid Khelil, Brahim Ayari, Piotr Szczytowski, and Neeraj Suri. Generic information transport for wireless sensor networks. In *Proceedings of the 3rd International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC)*, pages 27–34, 2010.

SimJava. The SimJava discrete event-based simulator. http://www.dcs.ed.ac.uk/home/hase/simjava, 2004.

Dale Skeen. Nonblocking commit protocols. In *Proceedings of the 1981 ACM SIGMOD International Conference on Management of Data*, pages 133–142, 1981.

Dale Skeen and Michael Stonebraker. A formal model of crash recovery in a distributed system. *IEEE Transactions on Software Engineering*, 9(3): 219–228, 1983.

D. Swaroop. *String Stability of Interconnected Systems: An application to Platooning in Automated Highway Systems*. PhD thesis, University of California, Berkeley, USA, 1994.

D. Swaroop and J.K. Hedrick. String stability of interconnected systems. *IEEE Transactions on Automatic Control*, 41(3):349–357, 1996.

Hung-Ying Tyan, Ahmed Sobeih, and Jennifer C. Hou. Design, realization and evaluation of a component-based, compositional network simulation environment. *Simulation*, 85(3):159–181, 2009.

Amin Vahdat and David Becker. Epidemic routing for partially-connected ad hoc networks. Technical Report Tech. Rep. CS-200006, Duke University, 2000.

Sudarshan Vasudevan, Jim Kurose, and Don Towsley. Design and analysis of a leader election algorithm for mobile ad hoc networks. In *Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP)*, pages 350–360, 2004.

Gary D. Walborn and Panos K. Chrysanthis. Supporting semantics-based transaction processing in mobile database applications. In *Proceedings of*

14th Symposium on Reliable Distributed Systems (SRDS), pages 31–40, 1995.

Gary D. Walborn and Panos K. Chrysanthis. Transaction processing in promotion. In *Proceedings of ACM symposium on Applied computing (SAC)*, pages 389–398, 1999.

Wanxia Xie. *Supporting Distributed Transaction Processing Over Mobile and Heterogeneous Platforms.* PhD thesis, Georgia Institute of Technology, Georgia, USA, 2005.

L. H. Yeo and Arkady B. Zaslavsky. Submission of transactions from mobile workstations in a cooperative multidatabase processing environment. In *Proceedings of the 14th International Conference on Distributed Computing Systems (ICDCS)*, pages 372 – 379, 1994.

Index

Curriculum Vitae

Personal Data

Name: Dipl.-Inform. Brahim Ayari

Date of birth: June 22nd, 1977

Place of birth: Tunis, Tunisia

School Education

1983-1989 Primary School, "École Primaire Khaznadar Denden", Tunis, Tunisia

1989-1996 High School, "Lycée Secondaire du Bardo", Tunis, Tunisia

University Education

1996-1997 *German Language Course* – Studienkolleg, Ruprecht-Karls-Universität Heidelberg, Germany

1997-1999 *Vordiplom in Computer Science* – Technische Universität Kaiserslautern, Kaiserslautern, Germany

1999-2004 *Diplom in Computer Science* – Technische Universität Kaiserslautern, Kaiserslautern, Germany

2004-2010 *Ph.D. in Computer Science* – Technische Universität Darmstadt, Darmstadt, Germany